Títol: Disseny i implementació d'un motor intel·ligent per el joc dels escacs

Volum: I

Alumne: Antonio Bagur Cantallops

Director: Javier Béjar Alonso

Departament: LSI

Data: 15/01/2009

Contingut

1. Introducció

El document que es presenta a continuació és la memòria del projecte de final de carrera titulat *Disseny i implementació d'un motor intel·ligent per al joc dels Escacs*.

L'objectiu d'aquest apartat és el d'introduir la temàtica del projecte i exposar les motivacions i objectius que han guiat la seva elaboració.

Així doncs, en el primer subapartat es presenten els objectius i motivació del projecte. En segon lloc, es presenta de manera genèrica el joc dels escacs. En el tercer apartat es parla dels antecedents i s'exposa de manera breu la història dels escacs per computador. Finalment, s'introdueix el context actual en el que es mouen els softwares d'escacs existents i la influència que aquets tenen dins del món dels escacs en general. Els lectors que sàpiguen que són els escacs i mínimament com s'hi juga es poden saltar el segon subapartat. Als que no, els hi recomano que llegeixin primer el subapartat 2 i després procedeixin a la lectura dels 1,3 i 4. Aprofito també per comentar que, donada la temàtica del projecte, apareixen durant el transcurs d'aquest text diversos termes molt específics que una persona no familiaritzada amb els escacs pot no entendre. Amb l'afany de que això no suposi un impediment per seguir el desenvolupament de les idees exposades s'ha annexat un glossari de termes al final del document al que es fa referència sempre que s'utilitza algun terme que pot ser desconegut per a persones no iniciades en el joc en general o, més en concret, als escacs per computador.

1.1 Objectius i motivació

La principal motivació per portar a terme aquest projecte surt de la meva afició per el món dels escacs en general, tant com a seguidor com participant actiu en tornejos. Cap al final dels meus cinc anys a la universitat i a causa de les ganes de millorar en el joc, vaig començar a utilitzar diferents programes d'escacs capaços de derrotar tots ells sense gaires esforços a jugadors que porten anys d'estudi, entrenament i participació en tornejos. Era tal la frustració experimentada en les successives derrotes al davant de l'ordinador que aviat hem vaig adonar de que segurament l'únic que hem podia proposar era entendre com s'engendra aquesta intel·ligència, en cap cas intentar derrotar-la, almenys a curt plaç. Va ser llavors quan vaig començar a documentar-me sobre les tècniques d'implementació de motors d'escacs[8] i a relacionar-les, si es que no ho havia fet abans, i contrastar-les amb tots aquells coneixements que havia après en les assignatures de la branca d'intel·ligència artificial.

Un cop convençut de que aquella temàtica m'interessava prou com per revalidar els meus estudis aprofundint en ella i donat que estava en consonància amb aquells coneixements que més havia gaudit aprenent i en els quals m'havia especialitzat mitjançant els perfils(les branques d'Intel·ligència Artificial i Enginyeria del Software) vaig redactar els objectius, a assolir amb el que es convertiria amb el meu Projecte de Final de Carrera, que detallo a continuació.

a) Estudiar l'estat de l'art de la implementació de sistemes intel·ligents que juguen a escacs i avaluar els pros i contres de cadascuna de les tècniques objecte de l'estudi.

Justificació: Aquest objectiu és fidel a l'aforisme *No reinventis la roda.* El escacs i les tècniques per implementar software que jugui de manera competitiva al joc dels escacs han estat estudiades àmpliament per la Intel·ligència Artificial, de fet s'estan estudiant des dels primers dies en que s'assentaren els fonaments de la mateixa. Sembla una bona elecció llavors intentar no reexplorar camins infructífers o intentar seguir i/o ampliar camins o tendències que han donat bon resultat.

b) Implementació d'un motor d'intel·ligència que jugui com a mínim a un nivell igual o superior al del 70% de jugadors de club, és a dir, que tingui un ELO major o igual a 1900.

Justificació: Si es fa una ullada ràpida a les llistes d'ELO que mesuren el rendiment dels motors d'escacs existents es veurà que amb aquesta condició no es garanteix pas que la majoria de motors existents no guanyin al que estem implementant. No obstant a això, cal tenir en compte que això és un mínim i no pas un màxim, és a dir, tot el que ens passem d'aquesta xifra serà ben arribat. El fet de que no s'hagi estat molt ambiciós a l'hora de fixar el rendiment mínim del motor intel·ligència ve donat pel fet de que es vol aconseguir un sistema ben dissenyat, modular i extensible que pugui servir com a marc de treball per a molts experiments d'algorismes aplicats als escacs. Per tal d'aconseguir això, s'han de sacrificar esforços d'altres llocs i un d'ells serà, de moment, la força efectiva que tenen altres motors d'escacs molt més monolítics.

c) Implementació del protocol UCI(Universal Chess Interface).

Justificació: Aquest objectiu, un cop assolit, permetrà que l'usuari pugui interactuar amb el motor d'intel·ligència mitjançant una interfície gràfica amigable com pot ser la oferta pel programa gratuït ARENA o la que equipa a la línia dels softwares comercials de CHESSBASE, per citar alguns exemples. Gràcies a això, s'obtindrà un gran valor afegit per a l'usuari amb un cost molt baix de desenvolupament el que permetrà focalitzar aquests guanys en el motor d'intel·ligència pròpiament dit.

La següent figura, il·lustra una partida home-màquina en la qual el jugador humà es comunica amb el jugador artificial mitjançant una interfície gràfica que es comunica amb el motor d'intel·ligència utilitzant el protocol UCI.

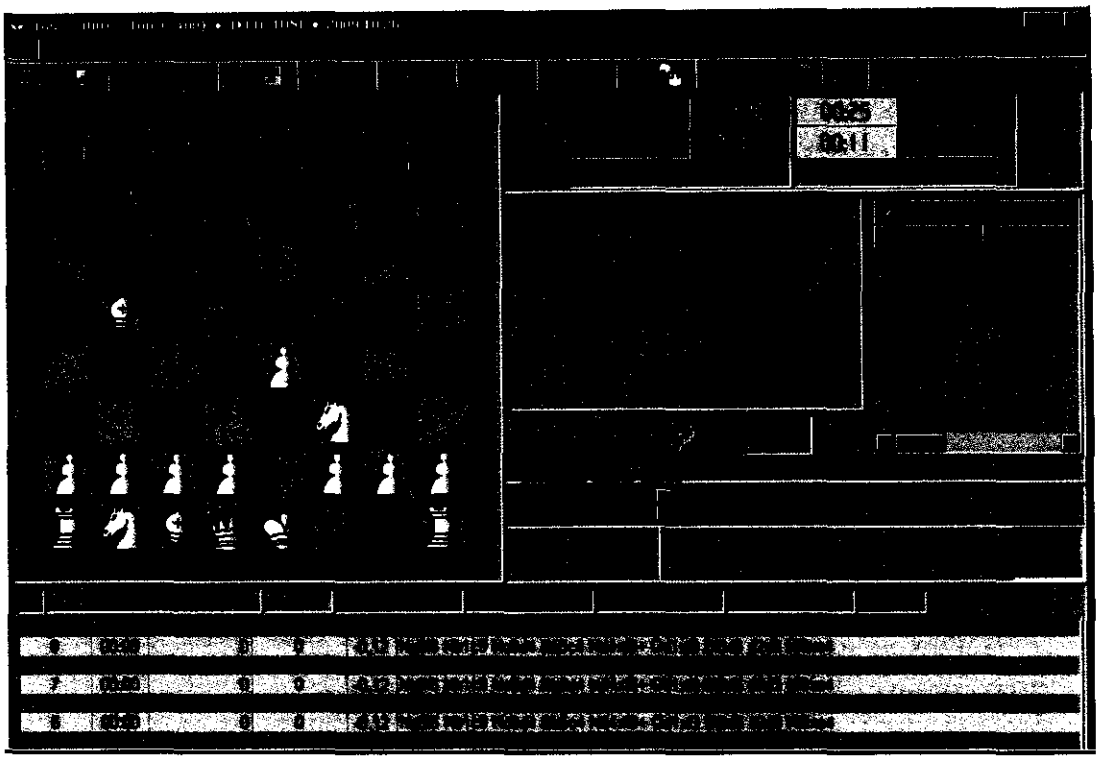


Figura 1: Maneig del motor d'inteligència desde una interfície d'usuari que implementa el protocol UCI.

1.2 El joc dels escacs

Els joc dels escacs és un joc de taula per a 2 jugadors molt conegut arreu del món. Més en concret, es classifica sovint com un joc de guerra descendent del *chaturanga*[9], del qual descendeixen també el *xiàngqí*(escacs xinesos) i el *shõgi*(escacs japonesos). Sovint es classifica també com un esport mental, una ciència i un art. Tant és així, que l'any 1994 el senat espanyol va recomanar el seu ensenyament en els col·legis declarant-lo ciència i esport. Arreu del món gaudeix també d'un reconeixement similar essent considerat disciplina esportiva a 156 països.

El joc es desenvolupa en un tauler quadrat de 8x8 caselles alternant-se caselles de color fosc amb caselles de color clar on s'hi col·loquen inicialment 16 peces per equip amb diferents capacitat de moviment. Cada equip disposa de un Rei, una Dama, dos Alfils, dos Cavalls, dos Torres i 8 Peons. Les peces acostumen a ser de color blanc per al primer jugador i de color negre per al segon. Es tracta d'un joc totalment racional en el sentit de que l'atzar no hi té una influència directa. Tot i que es desenvolupa en un espai finit amb un número finit de peces, s'estima que el número possible de partides d'escacs(conegut com número de Shannon) és major que el número total d'àtoms presents a l'univers. Aquest fet provoca que ni els millors mestres del món ni els programes més potents siguin capaços de calcular totes les contingències que es poden donar durant el transcurs del joc

El joc es desenvolupa alternant els torns d'ambdós jugadors on el que condueix les peces blanques disposa del primer donant-li un petit però significatiu avantatge ja que s'ha estimat a partir d'estudis en bases de dades que contenen milions de partides jugades per jugadors d'alt nivell, que les peces blanques guanyen un 55% de les partides que acaben amb victòria d'algun dels dos jugadors enfront al 45% de victòries amb les peces negres.

Una partida d'escacs pot acabar en taules(ja sigui perquè no queden suficients forces per guanyar la partida o bé perquè s'hagi arribat a un acord) o bé en victòria de un dels dos bàndols ja sigui perquè s'ha donat *escac i mat*[10] al rei contrari, perquè el rival abandona la partida o bé perquè se li ha acabat al temps. A part, des del 2005, el jugador al qual li soni el telèfon mòbil durant el transcurs de la partida també la perdrà immediatament. A diferència del que passa sovint en les partides d'aficionats, la majoria de partides que es guanyen no es guanyen per *escac i mat* sinó per abandonament del rival a causa de que la majoria de jugadors experimentats abandonen quan la derrota és imminent i òbvia.

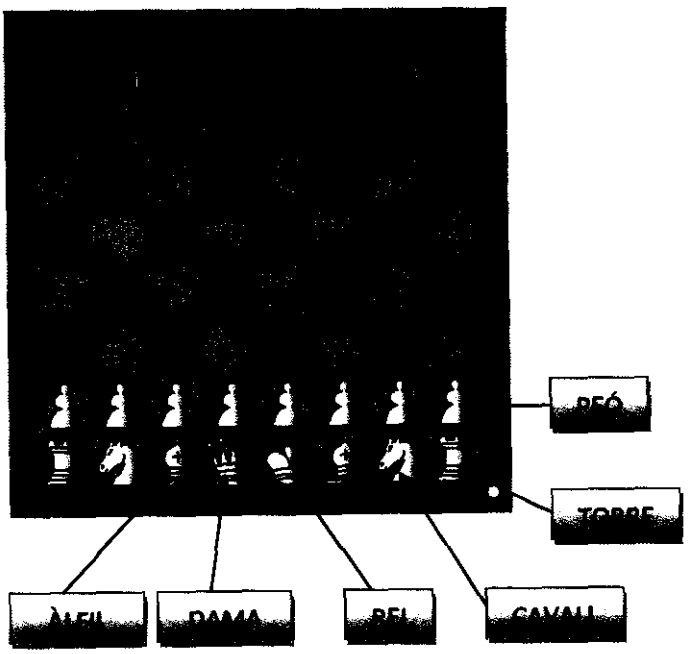
1.2.1 Regles dels escacs

1.2.1.1 Propòsit

L'objectiu central del joc per a cadascun dels dos jugadors és la captura del rei rival. La captura no s'arriba a materialitzar mai però quan el rei és atacat per el rival i no té manera de defensar-se de la captura que s'està amenaçant es diu que el rei està en escac i mat i s'acaba immediatament la partida, guanyant-la el jugador que ha fer l'escac i mat.

1.2.1.2 Començament del joc

El joc comença amb les peces col·locades de la següent manera:



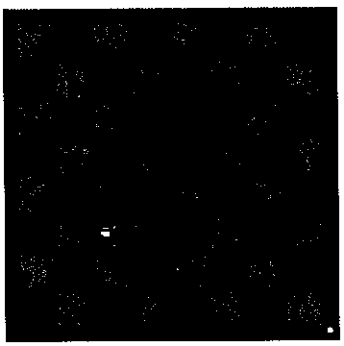
Inicialment, el primer moviment el fan les peces blanques i a continuació es van alternant el jugador negre i el jugador blanc fent un moviment cada cop que tenen el torn. Els jugadors no poden passar, és a dir, quan tenen el torn estan obligats a moure una i només una peça.

1.2.1.3 Jugant la partida

Un moviment consisteix en col·locar una peça en una casella diferent seguint les regles de moviment següents:

Rei

És la peça més valuosa de que es disposa. La pèrdua del rei suposa perdre la partida. Es pot moure en totes les direccions(horitzontal,vertical i diagonal) però només una casella per moviment(exceptuant el moviment especial d'enroc):

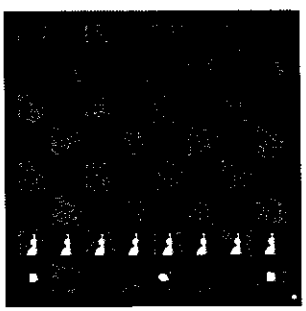


Si una de les caselles a les que es pot moure està ocupada per una peça rival no protegida la pot capturar.

L'enroc és un moviment especial on hi participen el rei i una torre que es pot portar a terme si es compleixen les següents condicions:

- El rei no s'ha mogut en tota la partida
- La torre que participa l'enroc no s'ha mogut en tota la partida
- El rei no es troba en Escac
- Totes les caselles entre la torre i el rei abans d'enrocar han d'estar buides.
- La casella situada entre la casella origen i la casella destí del rei no pot estar atacada
- La casella destí no pot estar atacada

Si es compleixen totes aquestes condicions es pot fer el moviment d'enroc que consisteix en que el rei es mou dues caselles horitzontalment en direcció a la torre i aquesta va a parar a la casella just després del rei(saltant-lo):

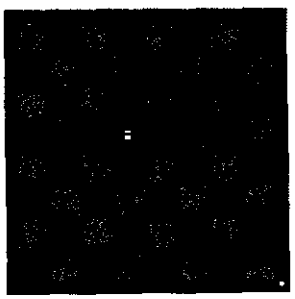


Dama

És la peça més poderosa. Pot moure's en totes les direccions(horizontal,vertical i diagonal) tantes caselles com vulgui. Com a contrapartida no pot saltar peces durant els seus moviments. Pot capturar les peces del contrari que toqui amb un sol moviment sense saltar peces seves ni del rival.

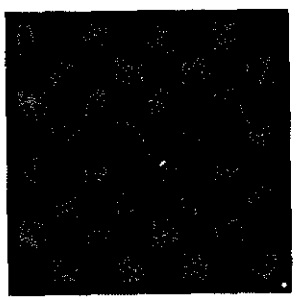
Torre

Es pot moure en sentit vertical i horitzontal tantes caselles com es vulgui. No pot saltar altres peces encara que sí pot capturar les peces del contrari que s'interposin en el seu moviment:



Alfil

Es mou en diagonal tantes caselles com es vulgui. A causa del seu tipus de moviment es mou sempre per caselles del mateix color:



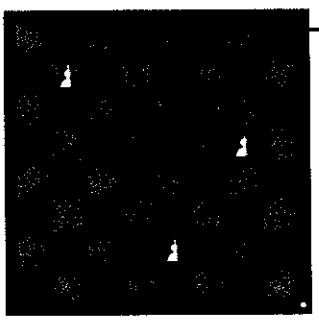
Cavall

És la peça que presenta el moviment més estrany podent saltar peces i movent-se en línies no rectes. Si considerem les files i les columnes del tauler eixos d'un sistema de coordenades en el pla, els moviments possibles que presenta el cavall corresponen a un dels següents vectors de desplaçament: (2,1),(1,2),(-2,1),(-1,2),(-2,-1),(-1,-2),(2,-1),(1,-2) que de manera gràfica es veuen així:

Peó:

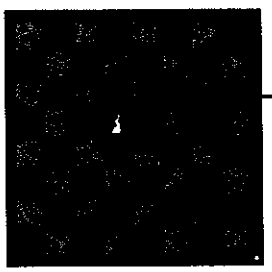
El peó presenta dos tipus de moviments: el d'avanç i el de captura. El d'avanç es fa sempre en direcció frontal i cap amunt respecte de la posició inicial del seu rei. En cas de que estigui en la seva casella d'origen pot avançar una o dues caselles. Des de les altres caselles pot avançar només una casella cap amunt. En el cas de que el peó arribi a la vuitena fila es transforma en Dama,Cavall,Alfil o Torre.

El moviment de captura es fa en diagonal i cap amunt(només una casella):



Promoció a Dama,Torre,Alfil o Cavall

Queda per comentar un moviment especial que és la captura en passant. Aquesta captura es dóna quan tenim un peó en cinquena fila i just en el torn anterior el contrari ha mogut un peó en una de les dues caselles que estan al costat del nostre peó. En aquest cas es pot capturar de la següent manera:



Únic cas en que es captura ocupant una casella diferent a la de la peça capturada. En aquest cas el peó negre desapareix.

1.2.1.4 Escac

Es diu que el rei està en escac quan una peça del bàndol contrari està amenaçant atacar-lo però pot escapar-se d'aquest atac. Existeixen tres maneres d'escapar-se de l'escac:

- Capturar la peça atacant
- Bloquejar la línia d'atac
- Escapar de la zona d'atac

El rei no es pot moure en una casella on estigui en escac i no es pot fer cap altre moviment que el deixi en escac(i.e. moure una peça pròpia que estava tapant l'atac d'una peça rival).

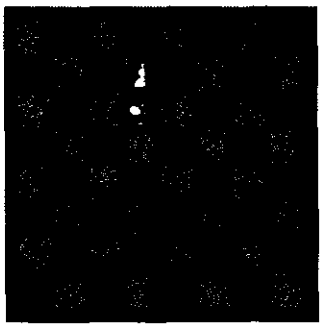
1.2.1.5 Escac i mat

Aquest és, en última instància, l'objectiu final de cada un dels jugadors: donar escac i mat al rei contrari.

Quan el rei està sota l'amenaça de ser capturat en la pròxima jugada i no es pot fer res per evitar-ho es diu que el rei està en escac i mat i es perd automàticament la partida.

1.2.1.6 Taules per ofegat

Aquesta situació es dóna quan el bàndol que té el torn no pot realitzar cap jugada legal i no es troba en escac. En aquest moment la partida es declara taules. La següent posició, en la qual els hi toca moure a les negres, és un exemple de tauler per ofegat:



1.2.1.7 Control de temps

Cadascun dels jugadors disposa d'un rellotge que comptabilitza el temps que li queda i que només corre quan és el seu torn. Si a algun dels dos jugadors se li acaba el temps perd la partida.

1.2.1.8 Victòria

Un jugador guanya la partida quan aconsegueix donar escac i mat al rei rival o bé quan el rival abandona.

1.2.1.9 Taules

La partida pot acabar en taules(empat) per un dels següents motius:

- Rei ofegat
- Acord entre els dos jugadors
- Final de partida de Rei + Rei
- Final de partida Rei + Rei + 1 peça menor(1 alfil o un cavall)

1.3 Història dels escacs per computador

En aquest apartat farem un repàs dels moments històrics més importants que han viscut els escacs per computador.

El primer intent documentat de crear un autòmat que jugués als escacs va aparèixer amb *El Turc*. Amb aquest nom es batejà a l'enginy introduït a Europa cap a finals del segle XVIII per part de l'inventor búlgar Wolfgang von Kempelen . Tot i que aquell enginy jugava molt bé, després d'uns quants anys de la seva presentació es va descobrir que era un frau i que les jugades, enlloc de fer-les l'autòmat, les feia un mestre d'escacs molt petit que s'amagava a dins de la màquina utilitzant una il·lusió òptica causada per la distribució dels seus compartiments.

Figura 2: Dibuix de l'autòmat anomenat *El Turc*

No fou fins el 1912 que no aparegué el primer autòmat que realment era capaç de jugar als escacs. En realitat només havia estat dissenyat per poder jugar un final de partida[11] de torre i rei contra rei però era capaç de guanyar aquest final fos quina fos la posició inicial encara que el seu joc no sempre era òptim(i.e. no donava escac i mat amb el mínim número de jugades teòric). A continuació es presenta una fotografia on es pot veure aquest invent que funcionava a partir d'un autòmat d'estats finits implementat amb mitjans mecànics i que movia les peces a partir d'un joc d'imans.

Figura 3: Fotografia de l'autòmat de Quevedo.

Fruit dels seus treballs pioners en el camp de la intel·ligència artificial, Alan Turing publica l'any 1947 la primera especificació d'un programa informàtic capaç de jugar als escacs. Dos anys més tard, Claude Shannon publica un article que descriu com programar una computadora per resoldre problemes de *mat en dues jugades*. Així mateix en aquest article proposa algunes millores per tal de restringir el número de combinacions possibles buscades per un programa que juga a escacs. Més tard, a l'any 1950, Turing escriu el primer programa complet per jugar a escacs encara que no és fins l'any 1956 que es veu un programa jugant a una versió reduïda dels escacs(tauler de 6x6) corrent a l'ordinador *MANIAC 1*.

L'any 1957 marca un punt d'inflexió ja que per primer cop es programa una computadora per que pot jugar una partida completa d'escacs. El seu creador va ser el programador d'IBM i jugadors d'escacs Alex Brenstein. El programa corria a sobre de la computadora IBM 704, una de les últimes que va funcionar amb vàlvules de buit.

Durant l'any 1958, per primer cop una computadora guanya a un humà en una partida d'escacs, la partida es va jugar contra un secretari al qual s'havia ensenyat a jugar a escacs una hora abans de la partida. Al 1959, uns quants programadors d'escacs pronostiquen que cap al 1970 el campió d'escacs serà una computadora, un enunciat força optimista per el que es va acabar donant en la realitat.

Al 1966, un programa d'escacs desenvolupat a la unió soviètica guanya a un altre desenvolupat a la universitat d'Standford a sobre d'un IBM 7090 en plena guerra freda, fet que no agrada gaire al govern d'Estats Units provocant que durant els pròxims anys inverteixi grans sumes de diners en aquesta àrea. Així l'any 1967 apareix ja el primer programa que guanya a un jugador d'escacs en el campionat de l'estat de Massachussets.

L'any 1968, el mestre internacional David Levy s'aposta 3000$ amb John McCarthy a que cap computadora el guanyarà amb 10 anys. Aquesta aposta la guanya David Levy.

L'any 1970 es funda el primer torneig d'escacs de computadores que guanyarà el programa CHESS 3.0. L'any 1971 l' *Institute of Control Science* de Moscou crea el programa KAISSA

utilitzant un computador anglès. Al 1974 aquest mateix programa guanya un torneig de computadores organitzat a Estocolm.

L'any 1976, CHESS 4.5 guanya la segona categoria del torneig Paul Masson de California. El seu ELO[1] promig fou de 1950.

Al 1977 apareix CHESS CHALENGER, la primera microcomputadora que juga a escacs i es funda la ICCA(International Chess Association) composta per uns 400 membres. Amb l'aparició el mateix any del programa CHESS 4.5 es dóna un altre salt qualitatiu ja que aquest programa guanya l'Open de Minnesota guanyant 5 de les 6 partides jugades i registrant una nova marca d'ELO promig pel que fa a computadores: 2271. En aquest torneig, Stemberg es converteix en el primer jugador de primera categoria en ser derrotat per una computadora. Aquest mateix any, Michael Stean es converteix en el primer Gran Mestre en perdre contra un ordinador.

A la dècada dels 80, també es batran noves marques: el programa HITECH arriba a un rating de 2530 l'any 1985 i l'any 88 apareix DEEP THOUGHT(Predecessor de DEEP BLUE) capaç d'analitzar 2 milions de posicions per segon guanyant en un match al mestre internacional David Levy. També s'enfronta contra Kasparov que guanya el match sense gaires problemes.

Durant la primera part de la dècada dels 90 els ordinadors guanyen molt terreny en les partides amb controls de temps curts(és sabut que com més curta és la partida, més avantatge té en principi l'ordinador): Fritz 2 i Fritz 3 guanyen a Kasparov en partides de 5 minuts i DEEP THOUGHT guanya a Judit Polgar en una partida de mitja hora.

Els problemes reals comencen amb l'aparició de DEEP BLUE, un software desenvolupat per IBM a sobre d'un supercomputador molt potent amb hardware dissenyat específicament per el domini dels escacs. El febrer del 96 s'hi enfronta Garry Kasparov, en aquell moment campió del món, que en aquesta primera ocasió guanya el match per 4-2. Tot i que Kasparov es salva de la derrota, és el primer cop que un campió del món perd contra un ordinador jugant en condicions de torneig normals.

L'any 97 Kasparov dóna la revenja a l'equip de programadors de DEEP BLUE i en aquesta ocasió no li va tant bé com la primera vegada, perd el match marcant un punt d'inflexió importantíssim : Per primer cop el campió del món d'escacs perd un match contra una màquina.

Des d'aquest moment fins a avui en dia, els programaris que juguen i analitzen partides d'escacs s'han consolidat com una eina indispensable per els jugadors d'alt nivell permetent que aquests es puguin entrenar contínuament contra un contrincant fortíssim i inesgotable, és a dir, s'ha donat per perduda la batalla i s'ha passat a veure l'ordinador com una eina d'entrenament. Segons el campió del món Kramnik, *la batalla home-màquina és una batalla perduda.*

2. Estat de l'art dels escacs per computador

2.1. Algorismes de cerca

Donat que, amb els coneixements que es tenen en l'actualitat, és impossible avaluar posicions de manera perfecte, els programes tenen que utilitzar algorismes de cerca per tal de poder jugar raonablement bé als escacs. Un algorisme de cerca consisteix en mirar algunes de les seqüències de possibles moviments futurs avaluant les posicions després dels moviments de les mateixes i concloent quina és la millor jugada.

Shannon, en els seus estudis pioners en aquest camp, va identificar dues famílies principals d'algorismes de cerca:

- **Algorismes de tipus A:** També anomenats algorismes de força bruta, la seva essència bàsica consisteix en explorar totes les variants existents en un arbre de cerca d'una certa profunditat.
- **Algorismes de tipus B:** Família formada per algorismes de cerca selectiva que exploren només les branques més prometedores de l'arbre de cerca.

Influenciats per els experiments fets pel psicòleg Adriaan de Groot, que empíricament serviren per fer una aproximació dels processos cognitius involucrats en l'anàlisi de posicions d'escacs, Shannon i altres investigadors pioners es van inclinar inicialment per els algorismes de tipus B. No obstant, des de l'aparició els anys 70 dels primers ordinadors raonablement potents, els algorismes de tipus A han dominat als de tipus B fins avui. Així, la majoria de programes existents en l'actualitat utilitzen l'estratègia de tipus A amb, això sí, algunes pinzellades de la filosofia dels algorismes de tipus B.

En aquest apartat es descriuran els algorismes més importants de la família de tipus A pel que fa a la cerca de la millor jugada respecte d'una funció d'avaluació donada, deixant per a l'apartat 2.3 algunes de les optimitzacions existents per a aquests algorismes.

2.1.1. Minimax

En el context de la teoria de jocs, l'algorisme Minimax és un sistema de presa de decisions en jocs amb adversari de suma zero(i.e. el guany d'un jugador és equivalent a les pèrdues del seu rival) sobre el qual és fonamenten algorismes més sofisticats. La seva utilitat principal és la de determinar la puntuació de cadascuna de les jugades disponibles en una posició donada després d'un determinat número de moviments considerant un joc perfecte dels dos adversaris.

A nivell teòric, si féssim una cerca exhaustiva no limitada per la profunditat arribaríem només a posicions terminals amb tres valors diferents: 1 victòria, 0 empat, -1 derrota. No obstant, el que passa a la pràctica és que els factors de ramificació mitjos dels jocs corrents són prou alts per a que les màquines més potents es quedin col·lapsades amb aquest tipus de càlcul fent necessària una solució menys ambiciosa. Davant d'aquesta necessitat, aparegué l'algorisme MiniMax amb profunditat limitada, les propietats del qual són les següents:

- Pertany a la família d'algorismes Depth First(Profunditat prioritària)

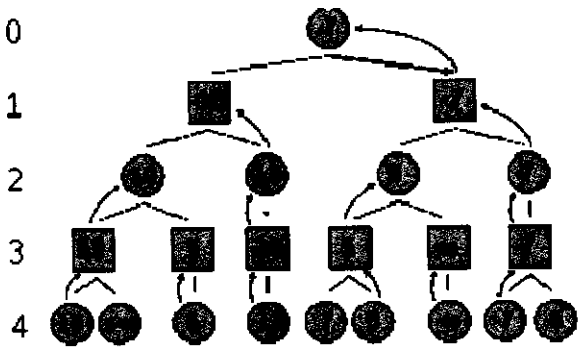
- Utilitza molt poc espai en memòria, essent aquest lineal respecte de la profunditat límit de la cerca(A l'apartat 2.3.1 es descriu una tècnica que permet aprofitar la gran quantitat de memòria RAM disponible en els ordinadors actuals).
- És molt costós en temps: la seva complexitat és exponencial respecte a la profunditat de la cerca, d'aquí que aquesta s'hagi de limitar per aconseguir la factibilitat de l'algorisme en la pràctica.

A continuació es descriu l'algorisme amb pseudocodi mitjançant dues funcions recursives que representen el comportament de cadascun del jugadors: Max(que intenta maximitzar la funció d'avaluació) i Min(que intenta minimitzar-la).

```
Funció Max(pos:POSICIO,prof:ENTER): retorna ENTER
    vars max,puntuacio:ENTER;
    fvars

    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna heurístic(pos);
    fsi

    max := -∞;
    percada fill ∈ fills(pos) fer
       puntuacio:= Min(fill,prof-1);
       si (puntuacio > max) llavors
          max:=puntuacio;
       fsi
    fpercada
    retorna max;

FFunció
```

```
Funció Min(pos:POSICIO,prof:ENTER): retorna ENTER
    vars max,puntuacio:ENTER;
    fvars

    si (prof=0)∨(EsTerminal(pos))  llavors
       retorna heurístic(pos);
    fsi

    min := +∞;
    percada fill ∈ fills(pos)fer
       puntuacio:= Max(fill,prof-1);
       si (puntuacio < min) llavors
          min:=puntuacio;
       fsi
    fpercada
    retorna min;
FFunció
```

El següent diagrama il·lustra un exemple d'execució de l'algorisme que acabem de descriure. Les rodones representen les decisions del jugador Max mentre que els quadrats representen les del jugador Min. Els valors de les fulles que es van propagant cap a l'arrel són els retornats per la funció heurística:



Donat que els escacs és un joc de suma zero, es compleix que heurístic(pos,puntDeVistaBlanques) = - heurístic(pos,puntDeVistaNegres).

Aprofitant aquesta propietat, es pot reescriure l'algorisme anterior utilitzant només una funció que es crida a ella mateixa coneguda popularment com NegaMax. El comportament d'aquest algorisme és idèntic a l'anterior amb la diferència de que la seva representació és més compacta. Notar que en la versió Minimax la funció heurística sempre retornava la puntuació des del punt de vista de Max(i.e peces blanques) i en canvi la versió NegaMax retorna la puntuació des del punt de vista del jugador que té el torn.

```
Funció NegaMax(pos:POSICIO,prof:ENTER): retorna ENTER
    vars max,puntuacio:ENTER;
    fvars

    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna heurístic(pos);
    fsi

    max := -∞;
    percada fill ∈ fills(pos) fer
       puntuacio:= -NegaMax(fill,prof-1);
       si (puntuacio > max) llavors
          max:=puntuacio;
       fsi
    fpercada
    retorna max;

FFunció
```

2.1.2. Algorisme MiniMax amb poda Alpha-Beta

Aquest algorisme és una millora substancial de l'algorisme MiniMax que presenta un estalvi significatiu per el que fa a la complexitat en temps de l'algorisme. S'estima que si l'algorisme minimax té una complexitat $O(b^n)$, l'algorisme alphabeta pot arribar a una complexitat òptima de $O(\sqrt{b^n})$. A més a més, aquest estalvi s'aconsegueix sense sacrificar de cap manera la fiabilitat de la puntuació resultant.

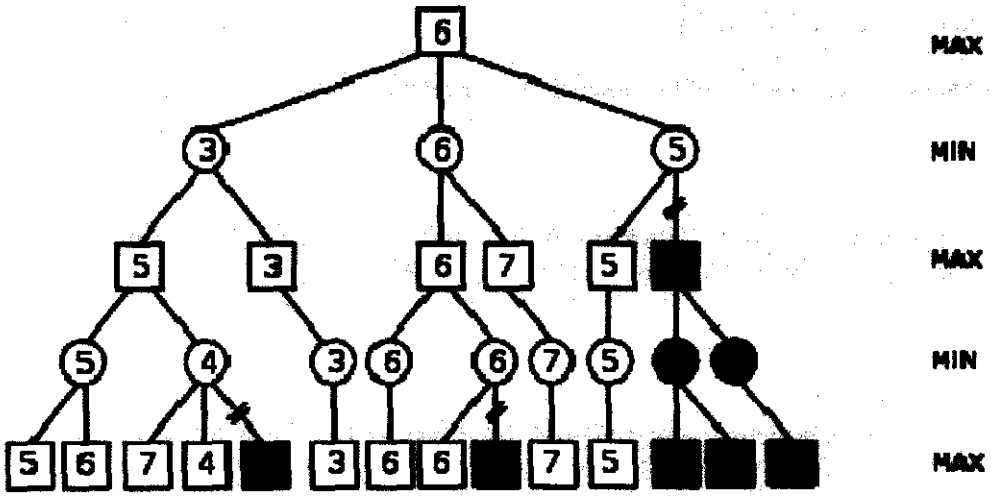
La seva idea principal és la següent: si tenim un moviment prou bo i estem buscant alternatives a aquest moviment, n'hi ha prou amb trobar una refutació per cadascuna d'aquestes alternatives, no cal trobar la millor refutació per a cadascuna d'elles.

L'estalvi real que faci de temps aquest algorisme vers l'algorisme Minimax depèn molt de la ordenació de jugades essent òptim quan s'exploren les millors jugades en primera instància. En el cas pitjor, quan es seleccionin les jugades en ordre decreixent de dolentes, l'algorisme trigarà el mateix que l'algorisme minimax. Evidentment no sempre serà possible explorar les millors jugades ja que si fóssim capaços d'això no caldria cap algorisme per saber quina és la millor jugada. De tota manera ajudaran molt a aquest comès les millores descrites als apartats 2.3.1, 2.3.5 i 2.3.6 augmentant significativament el rendiment d'aquest algorisme.

A continuació es presenta el pseudocodi corresponent a l'algorisme alphabeta utilitzant el marc de treball negamax per tal de que sigui més compacte en notació:

```
Funció AlphaBeta(pos:POSICIO,alpha:ENTER,beta:ENTER,prof:ENTER)
:retorna ENTER
    vars puntuació:ENTER;
    fvars
    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna heurístic(pos);
    fsi
    percada fill ∈ fills(pos) fer
       puntuacio:= -AlphaBeta(fill,-beta,-alpha,prof-1);
       si (puntuació ≥ beta) llavors
          retorna beta;
       fsi
       si (puntuació > alpha) llavors
          alpha:=puntuació;
       fsi
    fpercada
    retorna alpha;
FFunció
```

En la següent figura es pot veure un exemple d'execució de l'algorisme. Els quadrats representen nodes MAX i les rodones nodes MIN. Els nodes ombrejats en gris representen fulles/branques que no calen ser explorades a causa de que són podades. Notar que si es canviés l'ordre d'exploració les podes que es produirien serien unes altres:



2.1.3. Profunditat Iterativa

Des dels inicis de la investigació en algorismes de cerca, la tècnica de la profunditat iterativa va ser identificada com una tècnica robusta i funcional de gestionar el temps utilitzat per les cerques. La idea es ben senzilla, donat que a priori no se sap quan de complex serà l'arbre de cerca es comença buscant des de l'arrel de l'arbre a profunditat 1, després es busca a profunditat 2, després a profunditat 3 i així successivament fins que s'acaba el temps moment en el qual es retorna el resultat de la cerca més profunda del qual s'espera que serà el més fidedigne.

A priori, un dels defectes que aquesta tècnica pot tenir resulta obvi i és la pèrdua del temps invertit en fer cerques de profunditat inferior a la profunditat final a la que s'arriba. No obstant, aquest argument es pot refutar amb facilitat per els següents motius:

- Els algorismes utilitzats en la cerca són de complexitat exponencial. Suposem que el factor de ramificació mig(i.e. la base de la funció exponencial) és 50 i suposem també que s'ha arribat a una cerca de profunditat 10 que ha trigat 15000 ms. La següent taula conté una estimació del que hauria trigat cadascuna de les cerques a profunditat inferior:

Profunditat	**Temps(ms)**
9	300
8	6
7	0,12
6	0,0024
5	0,000048
4	0,00000096
3	1,92E-08
2	3,84E-10
1	7,68E-12
Suma	306,122449
	2%

  Tot i que no ho demostrem amb el rigor d'una demostració analítica, es pot veure que les cerques de profunditat inferior sempre representaran un percentatge molt petit del temps invertit en la cerca de profunditat màxima i que per tant és possible despreciar aquesta pèrdua de temps.

- En segon lloc, quan hem parlat de l'algorisme alphabeta s'ha vist que el rendiment d'aquest algorisme és òptim quan s'exploren primer els millors moviments. Gràcies a l'algorisme de profunditat iterativa i altres millores que discutirem més endavant com ara la taula de transposicions i el History Heuristic s'aconsegueix que l'ordenació de moviments sigui més òptima via l'aprofitament de la informació obtinguda en cerques de profunditat inferior.

El següent fragment de pseudocodi, il·lustra la mecànica seguida de l'algorisme de profunditat iterativa:

```
Funció ProfIterativa(): retorna ENTER
    vars valor,prof:ENTER;
    fvars
    prof:=1;
    fer
       valor:=buscar(prof);
       prof:=prof-1;
    mentre ┐TempsAcabat
    retorna valor;
FFunció
```

2.2. Representació de posicions

2.2.1. BitBoards

Una de les peces claus per a qualsevol software d'escacs és la representació de posicions, és a dir, el mètode o estructura que s'utilitza per representar una situació arbitrària de les peces en el taulell d'escacs juntament amb altre informació rellevant, com per exemple, el jugador al qual li toca moure en el torn actual. Existeixen varies alternatives per a representar posicions essent una de les més compactes i utilitzada avui en dia la tècnica dels BitBoards.

Aquesta tècnica es fonamenta en una representació centrada en les peces(enfront a altres que es centren en la casella) que utilitza N paraules de 64 bits. El fet de que el tamany de les paraules sigui 64 no és arbitrari sinó que coincideix amb el número de caselles que té un tauler d'escacs. Així doncs, cada paraula representa un subconjunt del conjunt finit format per totes les caselles del tauler. Un 1 a la posició *i* de la paraula indica que la casella pertany al subconjunt i un 0 indica el contrari. Havent arribat aquí es pot observar que el que necessitem per representar un tauler és una paraula per a cada tipus de peça *t* i color *c* de manera que la relació de pertinença de la casella *i* al subconjunt representarà el fet de que existeix una peça de tipus *t* i color *c* a la casella *i* del tauler. El següent exemple il·lustra millor aquest concepte servint-se de la posició inicial del joc:

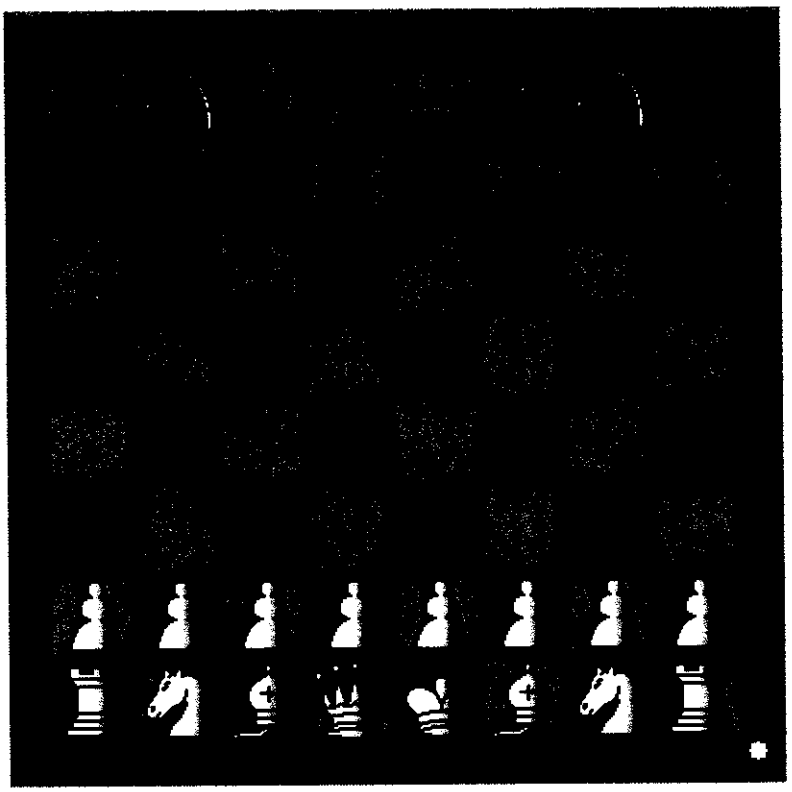


Figura 1: Posició Inicial en el Joc dels escacs

A continuació es detallen els valors que tindrien les paraules que representen a la posició anterior:

Representació de la posició inicial amb 12 BitBoards

Peons Blancs

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0

Cavalls Blancs

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0

Dames Blanques

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0

Peons Negres

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Cavalls Negres

0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Dames Negres

0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Torres Blanques

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1

Alfils Blancs

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0

Reis Blancs

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0

Torres Negres

1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Alfils Negres

0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Reis Negres

0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

2.3 Optimització de l'algorisme de cerca

2.3.1 Taules de transposició

2.3.1.1 Idea general

Aquesta tècnica s'utilitza per millorar el rendiment de la cerca. La seva idea fonamental és no buscar posicions que ja hem buscat i que es repeteixen a causa del fenomen de transposició. La transposició de posicions es dóna quan apareix una posició que ja havia aparegut abans amb una seqüència de moviments diferents. Per exemple la seqüència de moviments 1.e4 e5 2.Cf3 Cc6 porta a la mateixa posició que 1.Cf3 Cc6 2.e4 e5. Un algorisme de cerca com ara l'alphabeta sense utilitzar taula de transposicions busca una posició repetida com si no l'hagués buscat abans amb la disminució de rendiment que això suposa.

Per evitar les recerques de posicions que hem vist abans s'utilitza una taula gran indexada per un algorisme de hashing on es guarda per a cada posició que es busca una signatura, el valor de la posició buscada, la profunditat a la qual s'ha buscat, la millor jugada des d' aquesta posició, i un atribut que indica la precisió de la puntuació emmagatzemada. Quan anem a buscar una posició el primer que mirem és si ja ha estat buscada amb una profunditat superior o igual a la profunditat que estem buscant ara i en cas afirmatiu, es retorna el valor emmagatzemat en lloc de continuar la cerca.

A part d'evitar recerques inútils, aquesta tècnica també aporta beneficis addicionals als algorismes de la família alphabeta, on l'ordenació dels moviments és fonamental. Donat que per a cada posició emmagatzemem el seu millor successor, en la següent iteració de cerca(dins de l'algorisme de profunditat iterativa) el primer moviment que es provarà és el millor moviment de la iteració anterior.

2.3.1.2 Algorisme de Zobrist

Per tal de disposar d'una taula de transposicions, el primer que necessitem és un algorisme eficient de hashing que, donada una posició, ens retorni un índex on guardar la posició a la taula. L'algorisme més conegut per a aquest propòsit és l'algorisme de Zobrist. Aquest algorisme es pot dividir en dues fases: la fase d'inicialització i la fase de còmput de la clau. La fase d'inicialització només cal que s'executi un cop i consisteix en omplir una taula de paraules de 64 bits indexada per casella, tipus de peça i color de la peça de tamany 64*6*2=768 paraules. Cada casella d'aquesta taula s'inicialitza senzillament amb un número aleatori de 64 bits. Un cop tenim aquesta taula precalculada, en temps d'execució l'algorisme de càlcul de la clau actual és el següent:

```
Funció Zobrist(pos:POSICIO,taula:TAULA_ZORBIST)
: retorna PARAULA64BITS
    vars resultat:PARAULA64BITS;
    fvars
    resultat:=0;
    percada peça ∈ peces(POSICIO) fer
      resultat:=resultat XOR taula[tipus(peça),
                                   color(peça),
                                   casella(peça)];
    fpercada
    retorna resultat;
FFunció
```

Tot i que l'algorisme s'ha presentat en la seva versió total, donat que la funció XOR compleix la propietat que *a XOR b XOR b = a*, la clau de Zobrist es pot anar calculant de manera incremental a mesura que es van fent i desfent els moviments. Aquest algorisme té unes propietats que el fan ideal per al propòsit d'indexar posicions com per exemple el fet de que dues posicions molt semblants tenen claus de Zobrist totalment diferents.

Vist tot això queda un petit detall a comentar: amb una clau de 64 bits es genera un número de combinacions enorme i que, per tant, és impensable tenir una taula d'aquest tamany. El que es fa per arreglar aquest problema es retornar l'índex mòdul el número d'elements de la taula.

2.3.1.3 Escriptura de posicions

A l'hora de guardar la informació d'un node o posició que s'ha buscat, es poden donar tres casuístiques diferents:

a) Que el valor que es vol guardar sigui una fita superior del valor real de la posició: Aquest cas es dóna quan s'han explorat tots els fills de la posició i no n'hi ha cap que superi el valor del paràmetre alpha(això només es pot donar si estem treballant amb l'algorisme alphabeta o amb algún derivat seu com ara el PVS).
b) Que el valor que es vol guardar sigui el valor exacte de la posició. Aquest cas es dóna quan el valor retornat es major que el paràmetre alpha i menor que el paràmetre beta.
c) Que el valor que es vol guardar sigui una fita inferior del valor real de la posició: Aquest cas es dóna quan s'han explorat tots els fills de la posició i se n'ha trobat un el valor del qual supera el valor de beta.

El fet de que es puguin donar aquestes tres casuístiques ens obliga a que quan guardem la posició haguem de guardar un flag que indiqui de quin tipus de node es tracta.

El següent fragment de codi il·lustra les idees que hem discutit sobre l'escriptura de posicions:

```
Acció GuardaPos(pos:POSICIO,prof:ENTER,alpha:ENTER,beta:ENTER,valor:ENTER,
clau:PARAULA64BITS,taula:TAULA_TRANS,mmov:MOVIMENT)
    si(valor≤alpha) llavors
       taula[clau mod num_elems]:=element(pos,prof,alpha,mmov,F_SUPERIOR);
    sino si(valor<beta) llavors
       taula[clau mod num_elems]:=element(pos,prof,valor,mmov,EXACTE);
    sino
       taula[clau mod num_elems]:=element(pos,prof,beta,mmov,F_INFERIOR);
    fsi;
FAcció
```

2.3.1.4 Lectura de posicions

En aquest apartat discutirem la lògica algorísmica que hi ha al darrere de la recuperació de dades de posicions guardades a la taula de transposicions.

```
Acció RecordaPos(pos:POSICIO,prof:ENTER,alpha:ENTER,beta:ENTER,
clau:PARAULA64BITS,taula:TAULA_TRANS,sortida valor:ENTER,
sortida mmov:MOVIMENT)
vars prof2,valor2,mmov2,tip_node:ENTER;
fvars
   LlegirPos(clau,prof2,valor2,mmov2,tip_node);
   si(valor2 = NO_TROBAT) llavors
      valor := NO_TROBAT;
      mmov := NO_TROBAT;
   sino
      mmov := mmov2;
      valor:=NO_TROBAT;
      si(prof ≤ prof2) llavors
         si(tip_node=F_SUPERIOR ∧ valor2≤alpha) llavors
            valor:=alpha;
         fsi
         si(tip_node=EXACTE) llavors
            valor:=valor2;
            si(valor < alpha) llavor
                valor:=alpha;
            fsi
            si(valor > beta) llavor
                valor:=beta;
            fsi

         fsi
         si(tip_node=F_INFERIOR ∧ valor2≥beta) llavors
            valor:=beta;
         fsi
      fsi
   fsi
FAcció
```

2.3.1.5 Integració amb l'algorisme alphabeta

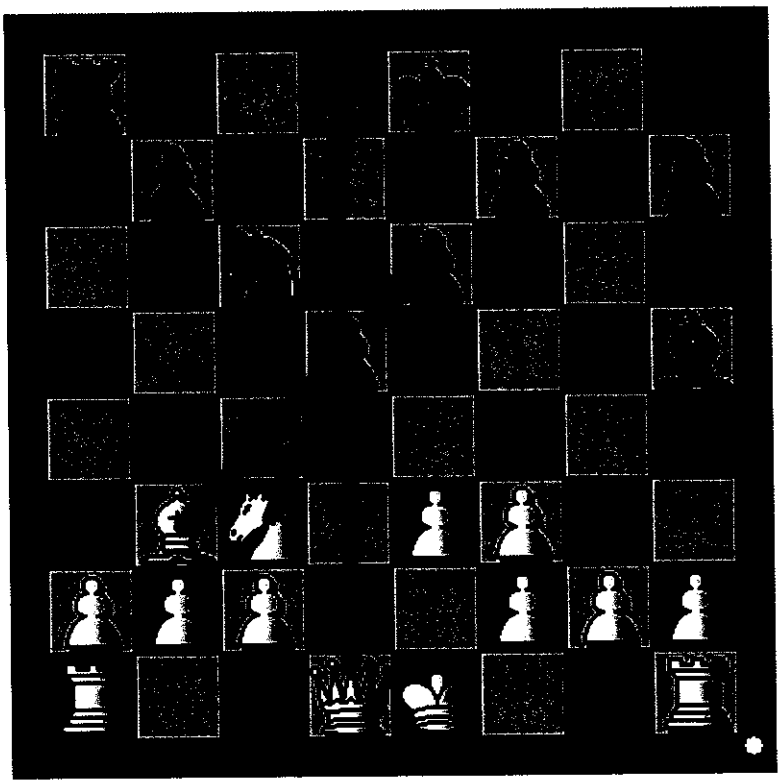
En aquest apartat anem a detallar la integració de la millora de la taula de transposicions amb l'algorisme alphabeta. El següent algorisme és l'algorisme alphabeta que ja hem vist amb les modificacions oportunes que es ressalten amb vermell per a una major claredat.

```
Funció AlphaBetaTT(pos:POSICIO,alpha:ENTER,beta:ENTER,prof:ENTER,
taula:TAULA_TRANS)
:retorna ENTER
    vars puntuacioMem,puntuacio,max:ENTER;
         mmov,mmov2:MOVIMENT;
    fvars
    RecordaPos(pos,prof,alpha,beta,Zobrist(pos),taula,puntuacioMem,mmov);
    si (puntuacioMem ≠ NO_TROBAT) llavors
        retorna puntuacioMem;
    fsi
    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna heurístic(pos);
    fsi
    generaFills();
    si(mmov ≠ NO_TROBAT) llavors
       ordenaFills(mmov);
    fsi
    mmov2:=mov(primer(fills(pos)));
    max:=alpha;
    percada fill ∈ fills(pos) fer
       puntuacio:= -AlphaBetaTT(fill,-beta,-max,prof-1);
       si (puntuacio ≥ beta) llavors
         GuardaPos(pos,prof,alpha,beta,beta,Zobrist(pos),taula,mov(fill));
         retorna beta;
       fsi
       si (puntuacio > max) llavors
          max:=puntuacio;
          mmov2:=mov(fill);
       fsi
    fpercada
    GuardaPos(pos,prof,alpha,beta,max,Zobrist(pos),taula,mmov2);
    retorna alpha;
FFunció
```

2.3.2 Avaluació estàtica dels intercanvis en una casella(Static Exchange Evaluation)

L'objectiu principal de l'algorisme que discutirem a continuació és el d'estimar com de bona és una captura de peça sense necessitat d'haver de fer una cerca exhaustiva del moviment. Aquest algorisme s'utilitza sovint durant la ordenació de moviments de captura per tal de poder donar una prioritat a cada captura, és a dir, saber quines captures tenen més possibilitats de ser les millors.

En la seva versió més senzilla, el que fa aquest algorisme es examinar la conseqüència de la millor sèrie d'intercanvis possibles en una casella determinada en un moment determinat i retorna el valor material a guanyar o a perdre. Suposem la següent posició:



És el torn de les blanques i, per saber quines captures són millors que les altres i poder ordenar els moviments de millor(estimació) a pitjor(estimació) s'aplica l'algorisme a cada moviment. Suposant per exemple que s'aplica al moviment "Dama pren peó de d5" l'algorisme retornarà un valor negatiu que indica que aquest moviment és un moviment perdedor ja que al moviment en qüestió segueix "Peó pren dama de d5" i després de "Cavall blanc pren a d5", les negres no estan obligades a capturar el cavall amb la dama i poden acabar amb la seqüència d'intercanvis.

En el següent requadre es detalla l'especificació de l'algorisme amb pseudocodi:

```
Funció SEE(pos:POSICIO,casella:CASELLA)
:retorna ENTER
    vars puntuació,val_see:ENTER;
         peça:PEÇA;
         fill:POSICIO;
    fvars
    puntuació:=0;
    val_see:=0;
    peça:=atacantMesPetit(pos,casella);
    si (peça = CAP_PEÇA) llavors
       retorna 0;{Ja no hi ha més atacs}
    fsi
    fill:=captura(pos,peça,casella);
    val_see := -SEE(fill,casella);
    si (val_see >= 0) llavors {Si perdem material no capturem}
       puntuacio:=peçaCapturada(pos,fill) + val_see;
    fsi
    retorna puntaucio;
FFunció
```

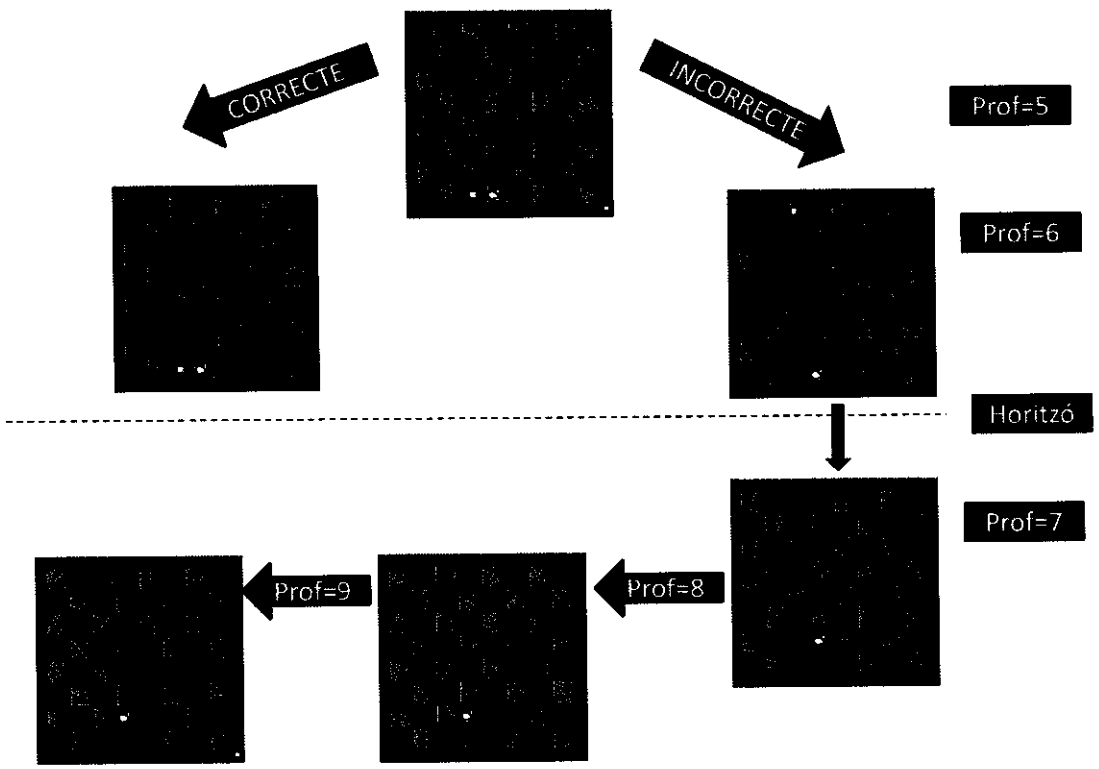
2.3.3 Cerca de posicions amb quietud tàctica (Quiescece search).

2.3.3.1 Problemàtica a resoldre

La principal aportació d'aquesta tècnica és la reducció dels efectes negatius que té un dels majors problemes dels algorismes de la família Minimax/Alphabeta(amb profunditat limitada):*l'efecte horitzó.*

Aquesta problema es dóna quan, per qüestions de factibilitat, es limita la profunditat de l'arbre de cerca i hi ha un canvi brusc en la funció d'avaluació just a la profunditat frontera. Suposem per exemple,en un exemple aplicat al domini dels escacs, que les blanques estan buscant una posició a profunditat 6 on inevitablement han perdut una dama. Suposem també que existeix una altra seqüència de moviments alternativa on, mitjançant el sacrifici d'una torre, posposen la inevitable captura de la dama fins a profunditat 8. A causa de l'efecte horitzó, l'algorisme arribarà a la conclusió que gràcies al sacrifici d'una torre ha evitat la captura de la dama quan en realitat el que fan es perdre les dues peces, la torre i la dama.

El següent diagrama il·lustra el problema que s'ha descrit:



2.3.3.2 Solució del problema

Una de les solucions típiques que es dóna a l'efecte horitzó és un algorisme anomenat *Quiescence Search.* En resum, la seva idea general és que, dins el marc d'un algorisme de cerca amb profunditat limitada, només s'aplicarà l'avaluació estàtica a posicions que es caracteritzin per una certa "quietud tàctica", és a dir, posicions des de les quals no es pugui, amb un sol moviment, capturar una peça rival ni promocionar un peó a una peça major i no estiguin amb escac. Així quan, per exemple, l'algorisme alphabeta arriba a la profunditat límit, enlloc de cridar directament a la funció d'avaluació crida a la funció *quiesce* la qual continuarà explorant les jugades no quietes recursivament i retornant la funció d'avaluació tant bon punt s'arriba a una posició terminal.

Tot i que la ramificació addicional és petita en comparació amb el factor de ramificació de posicions normal i de que amb pocs moviments les posicions no quietes convergeixen en posicions quietes, l'increment en el temps de cerca és considerable, el que provoca que s'hagin d'aplicar algunes podes a la tècnica per tal de que no siguin majors les pèrdues que els guanys.

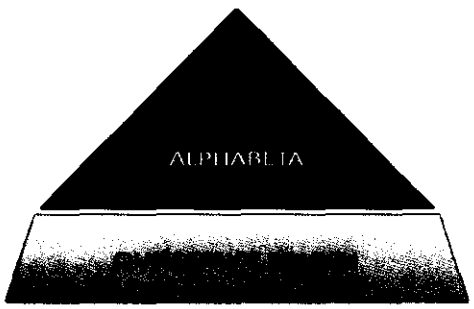


Figura 2: El creixement de l'arbre de cerca de la funció quiesce és més moderat que el de la funció alphabeta

A continuació es presenta una implementació senzilla de la funció quiesce:

Com es pot apreciar en l'anterior algorisme, el primer que es fa és calcular el valor de la funció

```
Funció Quiesce(pos:POSICIO,alpha:ENTER,beta:ENTER):retorna ENTER
    vars stand_pat,puntuació:ENTER;
    fvars
    stand_pat:=heurístic(pos);
    si (stand_pat >= beta ) llavors
       retorna beta;
    fsi
    si (alpha < stand_pat ) llavors
       alpha:=standpat;
    fsi
    percada fill ∈ fillsMovimentsTactics(pos) fer
       puntuacio:= -Quiesce(fill,-beta,-alpha);
       si (puntuació ≥ beta) llavors
          retorna beta;
       fsi
       si (puntuació > alpha) llavors
          alpha:=puntuació;
       fsi
    fpercada
    retorna alpha;
FFunció
```

d'avaluació estàtica i s'assigna a la variable *stand_pat*. Aquest terme s'agafa del poker i denota jugar una mà sense demanar més cartes. El que es busca amb aquesta variable és que el jugador que té el torn pugui decidir no capturar si la puntuació actual ja li està bé. Això ve de l'observació en anglès anomenada *null move observation* segons la qual s'estableix com a axioma que un jugador que tingui una bona posició regalant el torn a l'adversari la tindrà millor si mou. És sabut que aquesta norma té excepcions sobretot en posicions anomenades de Zugzwang[12].

A continuació es presenta una versió de l'algorisme alphabeta integrada amb la millora que s'acaba de descriure,ressaltant els canvis en vermell per a una major claredat:

```
Funció AlphaBeta(pos:POSICIO,alpha:ENTER,beta:ENTER,prof:ENTER)
:retorna ENTER
    vars puntuació:ENTER;
    fvars
    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna Quiesce(pos,alpha,beta);
    fsi
    percada fill ∈ fills(pos) fer
       puntuacio:= -AlphaBeta(fill,-beta,-alpha,prof-1);
       si (puntuació ≥ beta) llavors
          retorna beta;
       fsi
       si (puntuació > alpha) llavors
          alpha:=puntuació;
       fsi
    fpercada
    retorna alpha;
FFunció
```

2.3.4 Poda del Moviment Nul(Null Move Heurístic)

La tècnica que descriurem a continuació serveix per augmentar el número de podes beta durant la cerca de l'algorisme alphabeta. La idea és la següent: si el jugador al que li toca jugar, després de renunciar al seu torn i cedir-lo al rival, té una bona posició(i.e. el valor retornat per l'algorisme alphabeta realitza una poda) té una bona posició també sense cedir el seu torn. Aquest raonament ve d'una espècie d'axioma existent en el món dels escacs per computador conegut com la *Null move observation*. Aquesta observació diu que en la majoria de posicions, per al jugador al qual li toca moure sempre existeix una acció(jugada) millor que no fer res(existeixen posicions anomenades de Zugzwang[12] en el qual això no és així però donat que són poques i rares habitualment s'ignora aquest fet). El principal guany d'aquesta tècnica és que la cerca amb el moviment nul es fa amb una profunditat reduïda(normalment dos o tres nivells menys) de manera que la cerca és força econòmica en temps comparat amb les cerques dels moviments fills. El perquè de reduir la profunditat s'explica de la següent manera: el fet de cedir l'avantatge del torn es considera un avantatge prou gran com per reduir l'espai de cerca. El problema és que a les posicions on no es fa la poda la cerca addicional s'ha fet en va. De tota manera, s'ha comprovat de manera empírica que els guanys superen els costos en el cas mig. Tant és així, que la majoria de motors intel·ligents de codi obert existents incorporen aquesta tècnica.

A continuació es presenta el pseudocodi de l'algorisme alphabeta modificat amb aquesta millora(amb els canvis que suposa en vermell):

```
Funció
AlphaBeta(pos:POSICIO,alpha:ENTER,beta:ENTER,prof:ENTER,R:ENTER)
:retorna ENTER
    vars puntuació,puntuacioNul:ENTER;
         posNul:POSICIO;
    fvars
    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna Quiesce(pos,alpha,beta);
    fsi
    si(¬escac(pos)∧ prof ≥ 1+R) llavors
       posNul:=movimentNul(pos);
       puntuacioNul:=-AlphaBeta(posNul,alpha,beta,prof-1-R,R);
       si(puntuacioNul ≥ beta) llavors
          retorna beta;
       fsi
    fsi
    percada fill ∈ fills(pos) fer
       puntuacio:= -AlphaBeta(fill,-beta,-alpha,prof-1);
       si (puntuació ≥ beta) llavors
          retorna beta;
       fsi
       si (puntuació > alpha) llavors
          alpha:=puntuació;
       fsi
    fpercada
    retorna alpha;
FFunció
```

Típicament, el paràmetre R,que indica quan es disminueix la profunditat en el moviment nul, es fixa al valor 1 o al valor 2. Notar que, en cap cas, l'heurístic s'aplica si la posició actual està amb escac o bé la profunditat és inferior a la disminució de profunditat que s'ha de fer per aplicar-lo.

2.3.5 Heurístic del moviment assassí(Killer Heurístic)

Amb aquest nom es coneix a una tècnica dinàmica força efectiva d'ordenació de moviments que no són captura de peça. En la seva forma més comuna, la implementació d'aquesta tècnica consisteix en una taula de moviments(indexada per el nivell de profunditat en la cerca) i la següent estratègia d'actualització: quan es troba un moviment que provoca una poda beta es guarda aquest moviment en la taula descrita per al nivell de profunditat que li pertoqui.

Aquesta informació serveix després per ordenar els moviments de no captura donant una puntuació més alta als moviments que són *killer moves.* L'argument que hi ha al darrera d'aquest sistema d'ordenació de moviments és el següent: Existeixen moltes posicions en les quals hi ha tan sols un conjunt molt petit de moviments que crea un atac o es defensa d'ell. Tots els altres moviments que no ho fan, és molt probable que puguin ser refutats per el mateix moviment, el *killer move*.

No obstant, un problema no poc freqüent és que poden aparèixer amb certa freqüència moviments que no alteren la naturalesa del *killer move* però que s'han de tractar amb urgència deixant per més tard a aquest últim. Un exemple típic d'això és l'amenaça de la pròpia dama per part del rival. La solució que es dóna a aquest problema és, típicament, guardar dos ó tres, enlloc d'un, moviments per nivell muntant d'aquesta manera a cada nivell una espècie de cua circular de moviments de manera que si apareix un moviment que s'ha de tractar amb urgència no provoqui que el programa "oblidi" el *killer move* principal.

2.3.6 History Heuristic

2.3.6.1 Versió clàssica

Al igual que la que s'ha discutit a l'anterior apartat, aquesta tècnica és un mètode dinàmic d'ordenació de moviments que no són una captura de peça. A diferència de l'anterior, aquest es basa en el número de podes que ha causat un moviment amb independència del nivell(profunditat) en el que s'hagin fet. En la seva versió més senzilla, aquest heurístic manté una taula de puntuacions indexada per casella d'origen i casella de destí, fent una abstracció i depreciant el tipus de peça que hagi fet el moviment. Quan un moviment provoca una poda beta s'incrementa la posició corresponent a la taula. Un detall important és que normalment aquest increment és igual a *prof*prof* o bé 2^{prof} per tal d'aconseguir que pesin igual els moviments de l'arrel(d'alta profunditat però poc nombrosos) que els de les fulles(profunditat petita i molt nombrosos). A l'hora d'ordenar els moviments que no són captura s'utilitza aquesta puntuació donant més pes als moviments amb puntuació més alta.

Aquest sistema d'ordenar moviments és considerat una generalització, independent de la profunditat i la peça, del *killer heuristic.* A part, es conjectura que el contingut d'aquesta taula pot ser una bona representació abstracte dels plans a llarg plaç dels jugadors. No obstant aquestes consideracions, els últims anys ha perdut una mica de popularitat a causa de que els desenvolupadors dels motors d'escacs més punters han identificat que, quan s'arriba a

profunditats altes, aquesta tècnica introdueix una espècie de soroll aleatori en l'ordenació de moviments(cosa que abans no passava ja que la profunditat era molt més limitada).

2.3.6.2 Millora del History Heuristic: Relative History Heuristic

Un dels problemes o defectes que té el *History Heuristic* és el fet de que assumeix que tots els moviments apareixen amb la mateixa freqüència de manera que si això no és així en la pràctica prioritzarà o afavorirà els moviments amb una aparició més freqüent.

El *Relative History Heuristic* soluciona aquest problema. Bàsicament, el que fa és guardar una taula de puntuacions positives i una taula de puntuacions negatives. Un dels algorismes d'actualització més acceptats és el següent: Quan es dóna una poda beta s'incrementen els punts positius del moviment. En aquest mateix moment, per a cada moviment que s'havia provat en la posició amb anterioritat(no fent-ho per els que no s'han arribat a provar) s'incrementen els seus punts negatius. A l'hora d'ordenar moviments, la puntuació que s'utilitza és $\frac{puntsPositius}{puntsNegatius} * k$, on k és un coeficient que s'ha demostrat que amb valor ú dóna un rendiment força bo.

2.3.7 Principal Variation Search

L'algorisme que descriurem a continuació és una millora de l'algorisme AlphaBeta. La seva idea principal és que en la majoria de nodes no necessitem la seva puntuació exacte sinó que n'hi ha prou amb acotar el seu valor de manera que es pugui enunciar que el moviment és inacceptable per nosaltres o bé que el moviment és inacceptable per el nostre rival.

Necessitem només el valor exacte d'un moviment en el cas de que aquest pertanyi a la variant principal. Definim per variant principal a aquella seqüència de moviments acceptable per els dos jugadors, és a dir, aquella en la qual no es produeix cap poda beta en tot el camí i que es propagarà per tant des d'una fulla fins a la posició arrel.

Quan ja tenim el valor exacte de la variant principal d'un node, que en el cas d'utilitzar l'algorisme de profunditat iterativa obtenim el moviment a partir de la iteració anterior, assumim que els altres moviments no superaran aquest valor i per tant fem una cerca amb la finestra limitada , és a dir, en lloc de fer la crida recursiva amb [-beta,-alpha], com que suposem que el valor no superarà alpha, fem la crida recursiva dins la finestra [-alpha-1,-alpha]. El fet de reduir la finestra de cerca provoca que la cerca d'una variant no principal retorni més ràpid indicant-nos si estem amb lo cert, és a dir, que la variant actual no refuta a la variant principal. En cas de que estiguem equivocats i el valor retornat sigui major que alpha s'haurà de repetir la cerca amb finestra [-alpha,-beta].

Tot i les recerques, si l'ordenació de moviments és prou bona és major l'estalvi gràcies a refutar línies amb finestra limitada que el cost que suposen les recerques quan l'ordenació de moviments no ha estat bona.

A continuació es detalla el pseudocodi de l'algorisme *PVS* que acabem de discutir ressaltant amb vermell les diferències respecte a l'algorisme AlphaBeta:

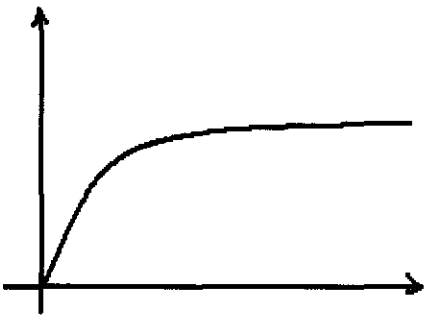
```
Funció PVS(pos:POSICIO,alpha:ENTER,beta:ENTER,prof:ENTER)
:retorna ENTER
    vars puntuació:ENTER;
         var_principal:BOOLEÀ;
    fvars
    si (prof=0)∨(EsTerminal(pos)) llavors
       retorna heurístic(pos);
    fsi
    var_principal:=CERT;
    percada fill ∈ fills(pos) fer
       si (var_principal) llavors
          puntuació:= -PVS(fill,-beta,-alpha,prof-1);
       sino
          puntuacio:= -PVS(fill,-alpha-1,-alpha,prof-1);
          si (puntuació > alpha) llavors
             puntuació:= -PVS(fill,-beta,-alpha,prof-1);
          fsi
       fsi

       si (puntuació ≥ beta) llavors
          retorna beta;
       fsi
       si (puntuació > alpha) llavors
          alpha:=puntuació;
          var_principal:=FALS;
       fsi
    fpercada
    retorna alpha;
FFunció
```

2.3.8 Paral·lelització de l'algorisme de cerca

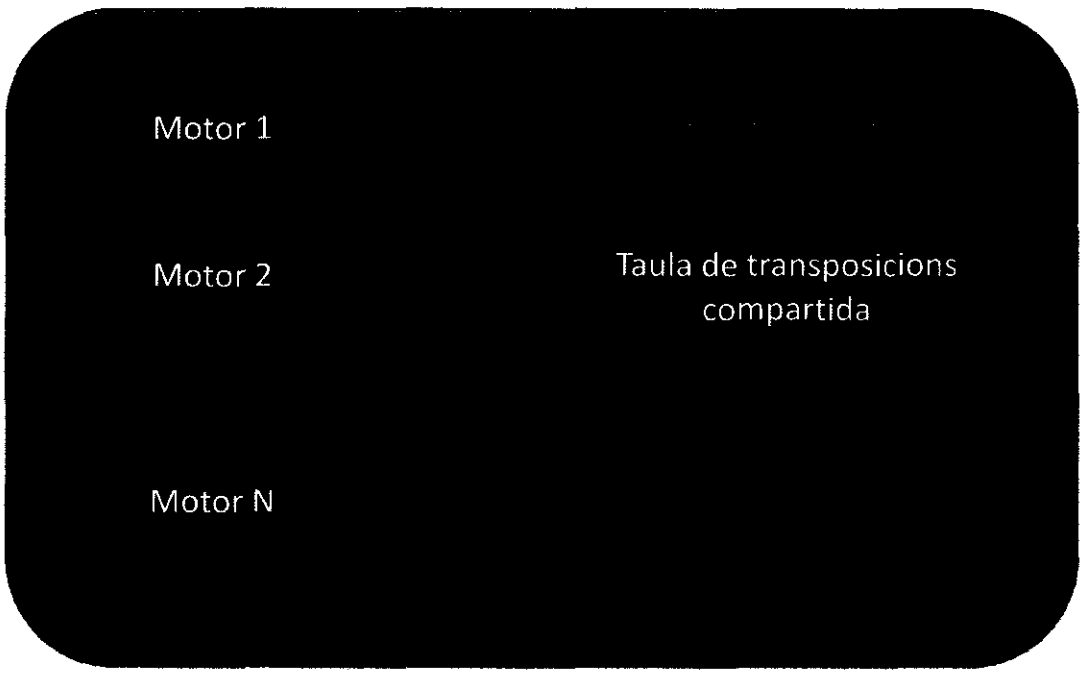
En aquest apartat es presenten les tècniques de paral·lelització de l'algorisme de cerca alphabeta(i derivats) més conegudes i rellevants.

2.3.8.1 Taula de transposicions compartida

Aquesta tècnica és una de les meves conegudes i una de les que comporta una implementació més senzilla. Com a contrapartida és també la tècnica que presenta una escalabilitat més dolenta, és a dir, arriba un punt en el qual afegir més processadors no augmenta el rendiment de la cerca. El següent gràfic, que representa la corba de rendiment respecte del número de processadors, il·lustra aquest concepte.



La idea de construcció és ben senzilla. Si disposem de N processadors que volem aprofitar per a l'algorisme de cerca, creem N motors de cerca que comparteixin la taula de transposicions i els posem a buscar la mateixa posició en paral·lel. Fruit del no determinisme les posicions buscades per cadascun dels motors de cerca(threads) cada cop convergiran més i per tant es crearà una sinèrgia en la qual cada un d'ells compartirà els resultats trobats i aprofitarà els resultats trobats per els altres.

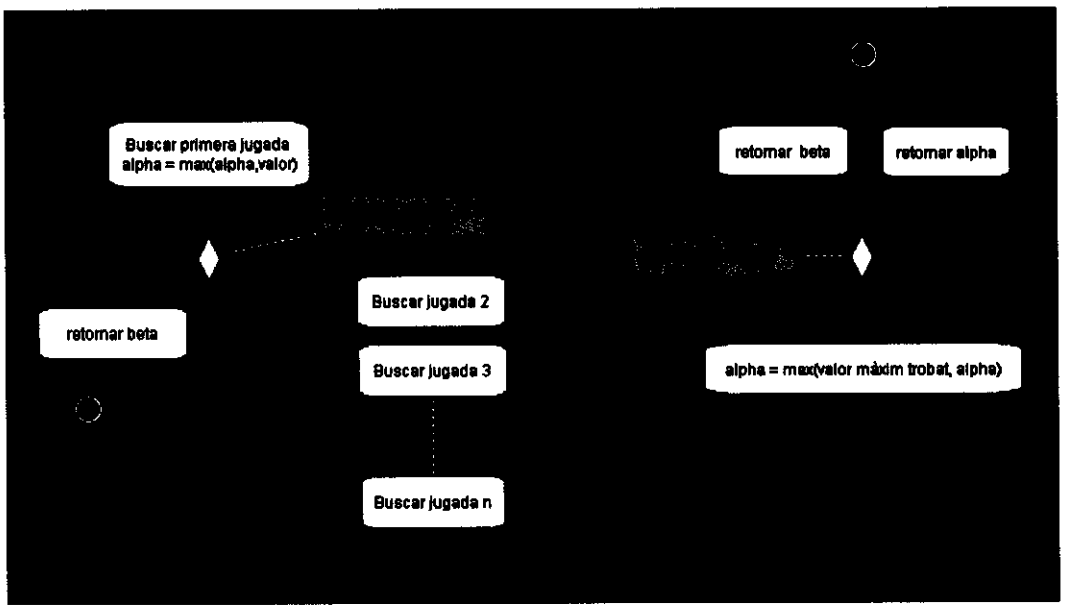


2.3.8.2 Young brothers wait

Aquesta tècnica, més eficient que l'anterior, es basa en el supòsit de que, dins el marc de l'algorisme de profunditat iterativa, la ordenació de moviments serà força bona i que per tant:

1) Existeixen altes possibilitats de que en la primera jugada es faci una poda
2) En el cas de que no es faci una poda és molt possible que la primera jugada retorni un valor més alt que alpha i que per tant serveixi com a una fita més restrictiva del paràmetre beta en el següent nivell.

Amb tot això, el que busca l'algorisme *Young Brothers Wait* és l'estalvi de temps invertit en la cerca via la següent tècnica que s'aplica a tots els nodes de manera recursiva: Es busca primer la jugada candidata a ser la millor(via la ordenació de moviments). Quan s'ha obtingut el valor d'aquesta si es supera beta la funció retorna beta. En cas contrari es llancen en paral·lel les cerques corresponents a la resta de jugades amb el paràmetre beta = - alpha(on alpha = max(alphaAnterior,valorTrobat)) . El següent diagrama d'activitat il·lustra la dinàmica que s'acaba de descriure.



2.4 Funció heurística d'avaluació de posicions

2.4.1 Introducció

La funció d'avaluació que utilitza l'algorisme de cerca serveix per mesurar el valor relatiu d'una posició en comparació amb altres i estimar quines són les possibilitats de guanyar des d'aquesta.

Si utilitzant l'algorisme de cerca es pogués veure el final de la partida per a cadascuna de les línies de joc n'hi hauria prou amb que aquesta funció ens retornés tres valors: -1 per a la derrota, 0 per a l'empat i 1 per a la victòria. No obstant, donada l'enorme combinatòria existent en l'arbre de possibilitats, s'ha de cridar a l'algorisme de cerca limitant-li la profunditat i per tant necessitem que la funció d'avaluació retorni un valor aproximat que ens serveixi per estimar les possibilitats de victòria des d'aquesta posició.

Serà en aquest punt on el coneixement serà més específic del domini dels escacs ja que moltes de les altres parts del motor d'intel·ligència són aplicables en principi a altres jocs amb adversari.

2.4.2 Morfologia de la funció d'avaluació

En la majoria de softwares existents, la funció d'avaluació de posicions consisteix bàsicament en una combinació lineal de diversos elements que es troben en una partida d'escacs. Aquets elements poden ser bàsics, com per exemple el número de peces de cada tipus o més complicats com per exemple determinar si el rei està segur en la seva posició o si determinada estructura de peons està en consonància amb la posició de l'alfil. Així doncs, si durant el disseny de la funció identifiquem N elements a tenir en compte, la funció d'avaluació serà similar a l'equació que presentem a continuació:

$Eval(pos) = \sum_{i=1}^{N} E_i * P_i$, on

E_i = 1 si l'element *i* es troba a la posició i 0 en cas contrari

P_i=Pes de l'element *i*

2.4.3 Principals elements que intervenen en l'avaluació de posicions

2.4.3.1 Factors materials de la posició

Quan els jugadors aprenen a jugar a escacs, inicialment avaluen les posicions fixant-se únicament en l'avantatge o desavantatge en número i tipus de peça. Així, i gràcies al que van fixar en el seu dia els mestres que es van dedicar a l'estudi del joc, es sap que una dama és més valuosa que una torre i que aquesta última és més valuosa que un cavall. En el context dels escacs, aquets elements que acabem de descriure es diuen *factors materials* ja que el que tenen en compte és bàsicament les forces materials de les que disposa cada jugador per atacar i per defensar-se. Quan s'avaluen posicions d'escacs amb ordinador, la unitat de mesura més freqüent per comparar posicions entre elles és el valor del peó, és a dir, s'assumeix que un peó val una unitat i a partir d'aquí es fixen els valors de les altres peces amb un número que

representa el valor relatiu de cada peça vers al peó. Normalment, la granularitat mínima de la funció d'avaluació és la centèsima de peó de manera que es pot enunciar per exemple que el valor d'un alfil es 3,10 vegades el valor d'un peó. Els valors que es poden apreciar a la següent taula són els valor relatius aproximats de les peces que s'accepten amb més freqüència encara que, com veurem més endavant, les peces poden tenir valors relatius en funció de la casella que ocupen:

Peça	Valor Relatiu en Peons
Cavall	Entre 3 i 3,30 en funció de l'autor
Torre	5
Rei	Infinit ja que si es perd el rei es perd la partida

2.4.3.2 Factors posicionals

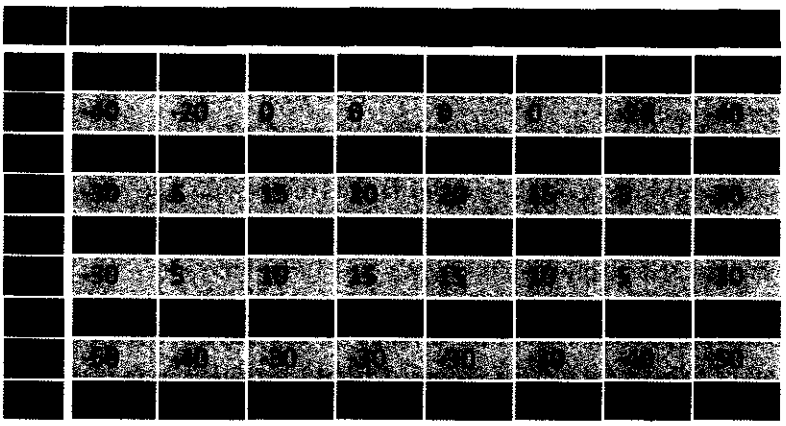
Tot i que els factors materials en una posició són fonamentals(i.e. una posició amb torre de menys es considera perduda) existeixen altres factors més subtils que poden servir per estimar també com d'avantatjosa és una posició vers a d'altres. Aquets factors s'anomenen factors posicionals i a continuació es presenten algunes dels més importants:

- **Estructura de peons:** A causa de la mobilitat reduïda que tenen els peons, qualsevol dany en la seva estructura(i.e. peons doblats[13] a causa d'una captura) pot esdevenir una debilitat permanent durant el transcurs de la partida ja que els esforços que s'hauran d'invertir per protegir la debilitat acabaran causant altres debilitats. A part de les debilitats i fortaleses que puguin causar és important també el fet de que l'estructura de peona condiciona molt també la mobilitat de les pròpies peces i les del rival. En definitiva, com va anunciar en el seu dia Phillidor, el peó és l'ànima dels escacs i la seva col·locació ha d'anar en consonància amb la resta de peces.
- **Seguretat del rei propi i atacs al rei rival**: L'objectiu final en els jocs dels escacs és el de capturar el rei rival. Donat que quan això passa es guanya la partida, el rei no té un valor propi com el de les altres peces sinó que es diu que és infinit. No obstant això, la posició d'aquesta peça i la posició relativa de les altres vers ella és molt important. Una funció d'avaluació mínimament forta haurà de ser capaç en definitiva d'avaluar la seguretat del propi rei i la del rei rival assignant els pesos oportuns a cadascun d'aquets aspectes.
- **Desenvolupament:** Aquest és un factor dinàmic de la posició que el que intenta és mesurar amb quina velocitat són capaces d'entrar en acció les peces de cada bàndol i quant de coordinades estan. És un concepte que té un pes important durant la fase d'obertura[14] del joc.
- **Peces atrapades**: En algunes posicions una peça es queda sense moviments possibles, ja sigui perquè està obstruïda per les peces pròpies o per les del rival(normalment peons). Quan això passa es diu que aquesta peça està atrapada i el seu valor efectiu passa a ser molt inferior al seu valor material teòric.
- **Mobilitat peces**: Aquest concepte és en certa manera l'antagònic de l'anterior. Com més possibles moviments tingui al seu abast una peça més valuosa serà aquesta ja

que podrà fer més funcions d'atac/defensa alhora. És per això que normalment s'intenta col·locar les peces en el centre, perquè allà disposen de més mobilitat.

2.4.3.3 Taules peça/casella

Una manera artificial però senzilla i pràctica d'introduir un cert *sentit posicional* a la funció d'avaluació són les taules peça/casella. Bàsicament són taules indexades per peça i casella que contenen per a cada casella i peça una petita puntuació que incentiva o penalitza que determinada peça estigui en una casella del tauler. Per exemple, si sabem que els cavalls treballen millor des de el centre del tauler que des dels extrems del tauler podem introduir una petita recompensa per el fet de que els cavalls estiguin en les caselles centrals. A continuació es presenta un exemple de taula peça/casella per al cas del cavall on es pot apreciar el que acabem de discutir(els valor són en centèsimes de peó):



2.4.4 Fase del joc

A part de tots els elements que hem vist a l'apartat 2.4.3, un altre aspecte molt important propi dels escacs que s'ha de tenir en compte és la fase de la partida. Una partida s'acostuma dividir en tres fases: obertura, mig joc i final de partida. Tots els elements que hem vist es veuen afectats d'una manera o altre per la fase de la partida. El rei per exemple és fonamental que estigui protegit durant les fases d'obertura i mig joc però en el final de partida en canvi ha de portar a terme un paper més actiu. Uns peons doblats poden no tenir influència en la fase d'obertura i ser en canvi determinants en el final de partida. És per això que una pràctica que s'ha popularitzat molt els últims anys és la de treure dues puntuacions: una puntuació de la posició segons els criteris de la fase d'obertura i una altra segons els criteris del final. A més a més, es calcula un coeficient, per estimar quant de propera està la partida del final, que varia dins d'un rang(entre 0 i 1 per exemple). Sota aquests supòsits, la puntuació final que s'assigna a la posició ve donada per la següent equació:

$$Eval(pos) = EvalObertura(pos) * \big(1 - fase(pos)\big) + EvalFinal(pos) * fase(pos)$$

2.4.5 Lazy Evaluation

Amb aquest nom es coneix a una tècnica que serveix per reduir el temps gastat en l'avaluació de posicions. Encara que la puntuació que aporten al global de la funció sol ser reduït amb comparació amb els elements materials, els elements posicionals acostumen a ser costosos de calcular. Aprofitant-se d'aquest fet, el que es fa és dividir la funció d'avaluació en dues fases: l'avaluació material(que és barata) i l'avaluació posicional. Quan es crida a la funció d'avaluació si els factors materials + k són menors que alpha es retorna alpha sense avaluar els factors posicionals i si beta + k és menor que els factors materials es retorna beta sense avaluar els factors posicionals; altrament, s'avaluen els factors posicionals i es retorna un valor més acurat. Que és llavors el paràmetre k? Per tal de no alterar el valor retornat per l'algorisme alphabeta ha de ser igual al màxim que pot augmentar o disminuir l'avaluació de la posició després de puntuar els factors posicionals.

3. Anàlisi de requisits

3.1. Requisits funcionals

A continuació es detallen els requisits funcionals que ha de complir el software per tal de que estigui alineat amb els objectius i motivacions del projecte i pugui, en conseqüència, ser acceptat.

Tipus de Requisit	Funcional	**Casos d'ús**	UC1,UC2,UC9,UC12,UC13

Descripció

El sistema haurà de ser capaç de jugar a una força de joc superior o igual a 1900 punts ELO[1]

Justificació

El fet de que sigui capaç de jugar a aquesta força de joc implica que serà capaç de guanyar al 70% de jugadors de club

Font Equip de desenvolupament/Projectista

Condició de satisfacció

El sistema guanya més d'un 90% de les partides disputades contra jugadors amb un ELO inferior o igual a 1900

Dependències	F3,F4	**Conflictes**	-

Material de suport

[1],[2],[3],[6]

Història

Creat el 20/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC3,UC4,UC5, UC6,UC7,UC8,UC9, UC10,UC11,UC12

Descripció

El sistema serà capaç de comunicar-se amb interfícies gràfiques que implementin el protocol de comunicació UCI[2](Universal Chess Interface)

Justificació

Els avantatges de complir amb aquest requisit són principalment 3. Per una banda, el fet de no haver d'implementar interfície gràfica ens permet centrar tots els esforços de desenvolupament únicament en la intel·ligència del programa. En segon lloc, el fet de que el programa parli el protocol estàndard UCI ens permet enfrontar-lo amb altres programes i mesurar així la seva força relativa. Finalment, ens permet reduir costos de desenvolupament i reaprofitar softwares excel·lents de domini públic com ara el programa Arena.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

Les funcionalitats del programa descrites en els requisits funcionals F3,F4 i F5 són accessibles via el protocol UCI

Dependències	F3,F5,U5,U7	**Conflictes**	-

Material de suport

[4]

Història

Creat el 20/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC2,UC9

Descripció

El sistema ha de permetre jugar partides amb control de temps estàndard ja sigui amb increment de temps per jugada o no. A cada jugada el programa rebrà via el protocol UCI el temps total que li resta dividint aquest i agafant-ne una part per calcular la jugada actual de manera que s'aprofiti bé el temps, és a dir, que no es perdin més d'un 1% de partides a causa de la caiguda de bandera[3] i que tampoc sobri més del 30% del temps en més d'un 20% de les partides perdudes.

Justificació

Per tal de que el sistema s'enfronti als seus adversaris amb una força de joc raonable la gestió del temps és fonamental. Per una banda, no es poden perdre partides a causa de que el programa es passi llargues estones calculant i se li acabi el temps, i per l'altre el sistema no pot contestar cadascuna de les jugades amb una petita fracció de temps que sigui constant. En definitiva, s'ha de gestionar el temps de manera adaptativa al temps restant.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

El sistema perd menys d'1% de les partides per caiguda de bandera i en un 80% de les partides perdudes li sobra menys d'un 30% del temps total que tenia la partida.

Dependències	F1,F2	**Conflictes**	-

Material de suport

[5]

Història

Creat el 21/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC2,UC12

Descripció

El sistema ha de permetre analitzar partides en mode "Anàlisi infinit"[4].

Justificació

Aquesta funcionalitat és fonamental de cara a que jugadors humans puguin entrenar-se mitjançant el programa i puguin també repassar les errades de les seves partides.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

El sistema és capaç d'analitzar la posició actual en mode "anàlisi infinit"

Dependències	F1,F5,U6,U7	**Conflictes**	-

Material de suport

[1],[2],[3],[6]

Història

Creat el 21/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC4,UC5, UC9,UC11,UC12

Descripció

El sistema ha d'informar regularment a la interfície gràfica de la següent informació: Línia principal buscada[5], número de nodes buscats, número de nodes per segon i profunditat actual de la cerca.

Justificació

La justificació d'aquest requisit és doble. Per una banda, gràcies a això l'usuari obtindrà de la interfície gràfica informació valuosa per a l'anàlisi de la partida en curs. En segon lloc, aquesta informació serà valuosa també per a l'equip de desenvolupament de cara a poder mesurar el rendiment del sistema.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

El sistema envia regularment la següent informació a la interfície gràfica: línia principal buscada,número de nodes buscats,número de nodes per segon i profunditat actual de la cerca.

Dependències F2,F4,U7 **Conflictes** -

Material de suport

[4]

Història

Creat el 21/02/2009

3.2. Requisits d'utilització i aparença

3.2.1 Introducció

En aquest apartat parlarem dels requisits d'utilització. Donat que, com s'ha vist a l'anterior apartat quan parlàvem dels requisits funcionals, el sistema implementa el protocol UCI, els requisits que llistarem a continuació no són requisits que ha de complir el nostre sistema sinó que són requisits que ha de complir el software que es connecti al nostre sistema.

3.2.2 Els requisits

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC3,UC5,UC9

Descripció

El sistema ha de presentar gràficament i clara la situació actual del tauler. Ha de quedar clar que és cada peça i la orientació del tauler(Si juguen blanques o negres).

Justificació

Encara que sigui un requisit obvi sovint no es compleix, sobretot en softwares dissenyats per a telèfons mòbils.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

El sistema presenta gràficament i clara la situació actual del tauler. És fàcil distingir el color i el tipus de cada peça i la orientació actual del tauler.

Dependències	U3,U5,U7	**Conflictes**	-

Material de suport

-

Història

Creat el 22/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC4,UC9

Descripció

El sistema ha de permetre jugar partides Jugador contra Màquina amb control de temps. Quan s'estigui utilitzant aquesta funcionalitat s'hauran d'ensenyar en tot moment el rellotge del jugador humà i el rellotge de la màquina.

Justificació

Quan es juguen partides amb control de temps el jugador necessita saber en tot moment el temps que li queda a cada bàndol.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

Quan s'està jugant una partida Jugador contra Màquina el sistema presenta en tot moment el rellotge dels dos bàndols.

Dependències	U7	**Conflictes**	-

Material de suport

-

Història

Creat el 22/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC3, UC9,UC10

Descripció

El sistema ha de permetre moure les peces mitjançant dos clicks (origen i destí) o bé mitjançant drag&drop (arrossegar les peces amb el ratolí)

Justificació

El moviment de les peces ha de ser còmode per a l'usuari ja que no tots els jugadors seran avançats i altres sistemes com per exemple introducció de notació algebraica serien massa complicats.

Font	Equip de desenvolupament/Projectista		
Condició de satisfacció			

El sistema permet moure les peces amb almenys un d'aquets dos sistemes: dos clicks(origen i destí) o bé mitjançant drag&drop.

Dependències	U1,U4,U5,U6,U7	**Conflictes**	-

Material de suport

-

Història

Creat el 22/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC3, UC9,UC10

Descripció

El sistema només ha de permetre a l'usuari fer moviments de peces que siguin una jugada correcte, mai moviments que esdevinguin una jugada il·legal.

Justificació

Tot i que segurament un usuari avançant cometi molt poques errades, les errades dels jugadors poc iniciats seran freqüents amb lo qual el sistema ha de poder detectar-les i no les ha de permetre

Font Equip de desenvolupament/Projectista

Condició de satisfacció

El sistema només permet fer un moviment a l'usuari sempre i quan sigui el seu torn i la jugada introduïda sigui una jugada legal.

Dependències	U3,U6,U7	**Conflictes**	-

Material de suport

-

Història

Creat el 22/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC4,UC9

Descripció

El sistema ha de permetre, a part de jugar i analitzar posicions des de la posició inicial, configurar el tauler amb una posició arbitraria i jugar i/o analitzar la posició configurada. És necessari a més que es comprovi que la posició configurada no sigui il·legal(i.e. que el jugador que no ha de moure estigui amb escac).

Justificació

Aquesta funcionalitat permetrà a l'usuari entrenar-se en les posicions que desitgi, com per exemple finals de partida típics.

Font	Equip de desenvolupament/Projectista		

Condició de satisfacció

L'usuari pot establir com posició actual qualsevol posició legal.

Dependències	F2,U1,U3	**Conflictes**	-

Material de suport

-

Història

Creat el 22/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC13,UC14, UC15,UC16

Descripció

L'usuari ha de poder introduir i analitzar qualsevol partida que hagi jugat

Justificació

Aquesta funcionalitat permet que l'usuari pugui introduir, per exemple, una partida que hagi jugat en un torneig i analitzar-la per poder detectar les errades que ha comès.

Font Equip de desenvolupament/Projectista

Condició de satisfacció

L'usuari pot introduir i analitzar qualsevol partida legal(jugada a partir d'una posició legal a partir de jugades legals)

Dependències	F4,U3,U4,U5,U7	**Conflictes**	-

Material de suport

-

Història

Creat el 22/02/2009

Tipus de Requisit	Funcional	**Event/Cas d'ús**	UC1,UC3,UC4 UC5,UC6,UC7, UC8,UC9,UC10, UC11,UC12

Descripció

La interfície gràfica ha d'implementar el protocol UCI[2] per tal de que s'integri bé amb el motor d'intel·ligència

Justificació

Aquest protocol és necessari a causa de que el motor d'intel·ligència l'utilitza.

Font	Equip de desenvolupament/Projectista		
Condició de satisfacció			
La interfície gràfica implementa el protocol UCI.			
Dependències	F2,F4,F5,U1,U2, U3,U4,U5,U6	**Conflictes**	-

Material de suport

-

Història

Creat el 23/02/2009

3.2.3 Software recomanat

Havent vist els requisits que ha de complir la interfície gràfica que s'instal·li amb el motor d'intel·ligència, en aquest apartat procedirem a recomanar alguns softwares existents que implementen el protocol UCI i compleixen els requisits esmentats. De totes maneres, és possible instal·lar el motor d'intel·ligència amb qualsevol software que compleixi els requisits i implementi el protocol UCI[2].

Així doncs la recomanació principal que es fa és utilitzar la interfície gràfica gratuïta Arena ja que no suposa despeses en llicències i compleix amb tots els requisits d'usabilitat esmentats. En cas d'usuaris avançats que necessitin funcionalitats avançades de gestió de bases de dades de partides es recomanaria una interfície gràfica comercial com la que ofereix el fabricant ChessBase® o la que ofereix el fabricant Chess Informant®.

3.3. Requisits de rendiment

Parlar de rendiment en un software d'escacs és difícil ja que no existeix, de moment, una fita superior. Aquest fet passa a causa de que, a diferència d'altres jocs com les dames, no es coneix una estratègia perfecte que porti sempre a la victòria o a l'empat, és a dir, de moment no es sap la resposta a la pregunta de qui guanya la partida des de la posició inicial, si és que es pot forçar la victòria(Podria ser que des de la posició inicial la partida sigui taules i no pugui forçar la victòria cap dels dos bàndols).

Ens haurem de conformar llavors amb mesurar el rendiment del sistema enfront d'altres sistemes que juguin a escacs(ja siguin artificials o no). D'aquesta manera, els requisits de rendiment ja queden coberts amb el requisit funcional número #F1. Algú podria objectar no obstant que queda per definir alguna restricció en el temps de càlcul però això no és cert ja que quan es parla de rendiment en partides ja es té en compte que aquestes són amb control de temps i que per tant si el sistema triga massa en computar les jugades perdrà senzillament per caiguda de bandera[3].

3.4. Requisits de hardware i sistema operatiu

En el punt anterior hem parlat de requisits de rendiment. No obstant, per tal de que el rendiment sigui mesurable, hem de restringir l'entorn on s'executarà ja que pot passar que els requisits de rendiment es compleixin en un supercomputador però no es compleixin quan el mateix software s'executa en la CPU d'un telèfon mòbil. Així en lloc de dir que el requisit de rendiment #F1 s'ha de complir, direm que el requisit de rendiment #F1 es complirà sempre i quan es compleixin o es superin els requisits de hardware que per el cas seran els següents:

- 1 CPU Pentium III o superior a una freqüència de rellotge de 750Mhz
- 1024MB de memòria RAM
- Targeta Gràfica que sigui capaç d'executar un entorn gràfic amb finestres tipus Windows, Gnome o KDE.
- Un sistema operatiu modern que sigui capaç d'explotar el hardware anterior amb eficiència com per exemple Linux amb Kernel 2.4 o superior, Microsoft Windows XP o bé Mac OS X.
- Un requisit addicional, donat de que el codi font del programa s'ha desenvolupat en C++, és que existeixi un compilador de C++ pel sistema operatiu escollit. De totes

maneres, això no hauria de ser un problema ja que el llenguatge C++ està molt extès i existeixen compiladors per a totes les plataformes citades.

Si els requisits anteriors no es compleixen al 100%, pot ser que el sistema segueixi funcionant sense problemes encara que no es garanteix que ho faci d'acord al requisit de rendiment #F1.

4. Especificació del sistema

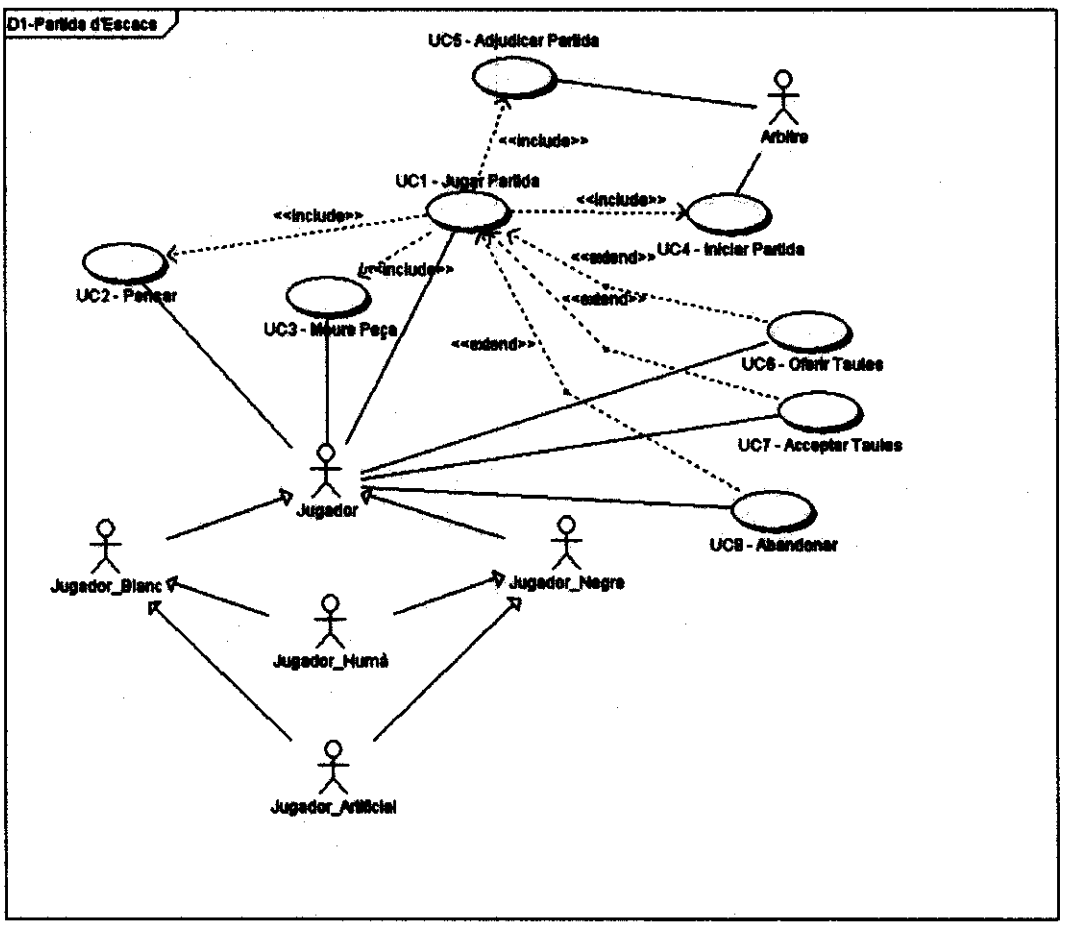
4.1 Especificació de casos d'ús

4.1.1 Diagrames de casos d'ús

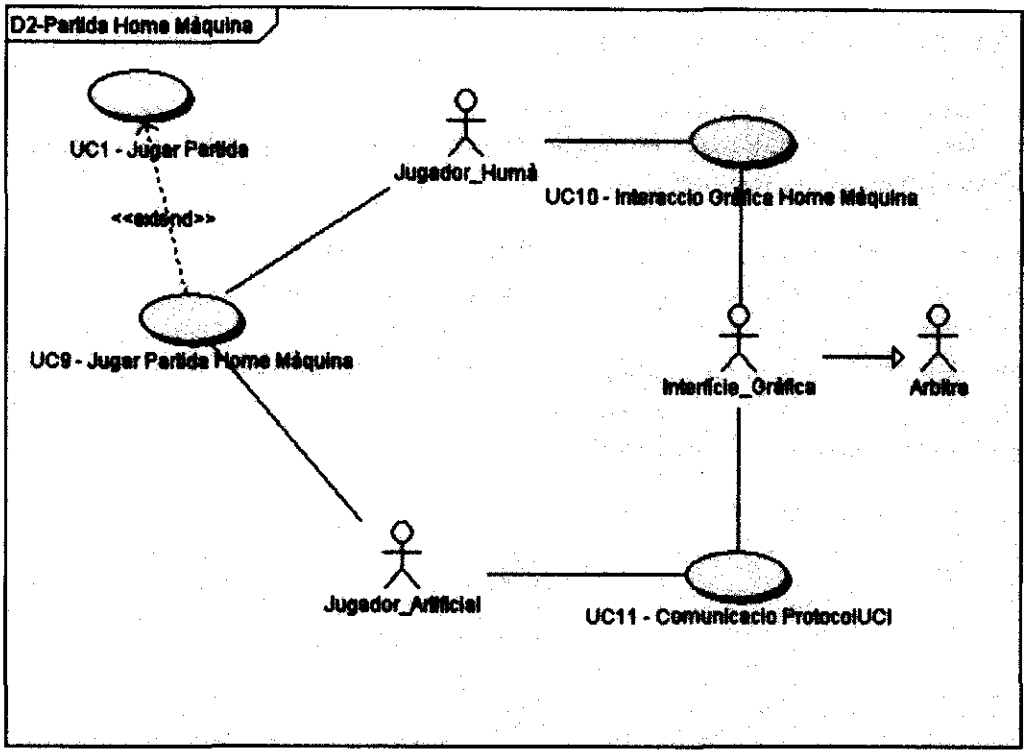
En aquest apartat presentarem els diagrames dels principals casos d'ús que es deriven dels requisits exposats en l'apartat 3.

Per a una major claredat, s'han dividit els casos d'ús en tres vistes diferents. En primer lloc, s'ha creat un diagrama dels casos d'ús involucrats en una partida genèrica d'escacs ja sigui home contra màquina o home contra home. En segon lloc, s'ha creat una segona vista dels casos d'ús que es donen en el cas particular de que la partida sigui home contra màquina. Finalment, se n'ha creat una tercera que captura els actors i la casuística involucrada en l'anàlisi retrospectiu[6] de partides d'escacs.

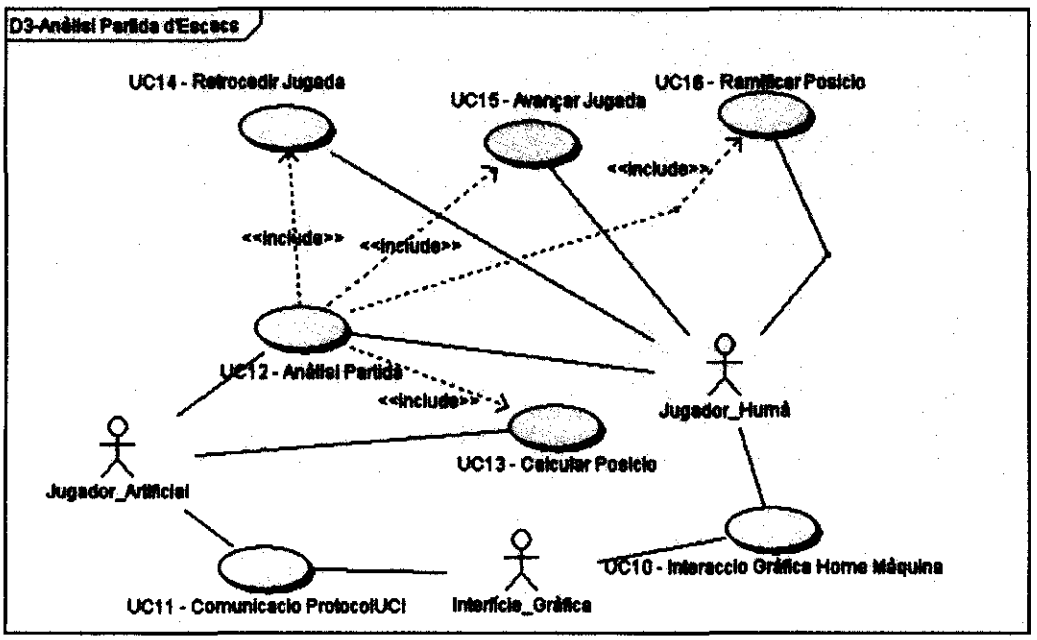
4.1.1.1 Casos d'ús Partida d'Escacs



4.1.1.2 Casos d'ús partida home – màquina



4.1.1.3 Casos d'ús relacionats amb l'anàlisi de partides d'escacs



4.1.2 Descripció dels casos d'ús

A continuació es presenta una descripció més detallada dels casos d'ús que hem vist a l'anterior apartat:

Cas d'ús: Jugar Partida **Actors:** Jugador

Propòsit: Jugar una partida d'escacs **Tipus:** Primari - Essencial

Resum:

Dos jugadors, un de peces blanques i l'altre de peces negres, s'enfronten davant d'un tauler amb el propòsit de guanyar la partida. Per a aquest propòsit, alternant-se jugador blanc i jugador negre, es van intercanviant les jugades que creuen millors en cada moviment.

Requisits: F1,F2,F3,F5,U1,U2,U3,U4,U5,U7

Curs típic d'esdeveniments:

	Actor 1: Jugador Blanques		Actor 2: Jugador Negres	Actor 3: Àrbitre
				1.L'àrbitre inicia la partida
Mentre la partida no s'hagi acabat	2. Les blanques pensen quina és la millor jugada possible.			
	3. Un cop decidida la millor jugada, les blanques fan el moviment que creuen millor.			
		Si no s'ha acabat la partida	4. Les negres pensen quina és la millor jugada possible.	
			5. Un cop decidida la millor jugada, les negres fan el moviment que creuen millor	
				6. Si hi ha guanyador, l'àrbitre li adjudica la partida. En cas contrari, la declara taules

Cursos alternatius:

Durant la partida es poden donar circumstàncies especials que desencadenin queixes i apel·lacions per part dels jugadors. De totes maneres aquest tipus de comportament només té lloc en les partides entre humans i per tant es queda fora de l'abast d'aquest document.

Cas d'ús:	Pensar(Jugada)	**Actors:**	Jugador
Propòsit:	Estimar quina és, donada la posició actual, la millor jugada possible	**Tipus:**	Primari - Essencial

Resum:

El jugador al qual li toca moure, pensa abans de fer-ho quina és la millor jugada possible donada la posició actual

Requisits: F1,F3,F4

Curs típic d'esdeveniments:

La descripció de tots els processos que es desencadenen quan el qui pensa és un ésser humà podria ocupar diversos llibres. Per el cas de jugadors artificials el flux més típic és buscar la posició actual durant un temps que varia en funció del temps que li resta per acabar la partida, decidint que la millor jugada és la primera jugada de la variant principal que s'ha trobat.

Cursos alternatius:

Quan es tracta d'un jugador artificial que utilitza llibre d'obertures, si la posició actual es troba al seu llibre, es decideix que la millor jugada és una de les que marca el llibre com a bones jugades. Anàlogament, si s'utilitzen taules de finals i la posició actual es troba a la taula de finals, es decideix que la millor jugada és la millor jugada que es troba a les taules de finals.

Cas d'ús:	Moure peça	**Actors:**	Jugador
Propòsit:	Fer el moviment que es creu que és millor	**Tipus:**	Primari - Essencial

Resum:

Després d'haver pensat, el jugador al qual li toca moure, fa el moviment que creu que és millor.

Requisits: F2,U1,U3,U4,U7

Curs típic d'esdeveniments:

En el cas de jugadors humans, es fa el gest mecànic de moure la peça (en el cas d'enfrontar-se a un jugador artificial ho fa via la interfície gràfica per drag&drop). En el cas d'un jugador artificial, comunica la jugada a la interfície gràfica mitjançant el protocol UCI.

Cursos alternatius:

-

Cas d'ús #UC4			
Cas d'ús:	Iniciar Partida	**Actors:**	Àrbitre
Propòsit:	Donar permís als jugadors per a que iniciïn la partida	**Tipus:**	Primari - Essencial
Resum:			
L'àrbitre engega els rellotges marcant el començament de la partida			
Requisits:	F2,F5,U2,U5,U7		
Curs típic d'esdeveniments:			
En el cas en que la partida sigui entre dos jugadors humans, l'àrbitre posa els rellotges en marxa i marcant el començament de la partida. En el cas en que la partida sigui home – màquina o màquina – màquina, la interfície gràfica assoleix el paper de l'àrbitre. En aquest últim cas, la partida és iniciada per un jugador humà que dóna la ordre a la interfície gràfica.			
Cursos alternatius:			
-			

Cas d'ús #UC5			
Cas d'ús:	Adjudicar Partida	**Actors:**	Àrbitre
Propòsit:	Decidir, un cop acabada la partida, qui ha guanyat(si és que la partida no acaba en empat).	**Tipus:**	Primari - Essencial
Resum:			
Quan la partida s'acaba, l'àrbitre decideix sí la partida la guanyen les blanques, les negres o bé és empat.			
Requisits:	F2,F5,U1,U7		
Curs típic d'esdeveniments:			
Quan la partida s'acaba, l'àrbitre decideix sí la partida la guanyen les blanques, les negres o bé és empat. En el cas en que la partida sigui home – màquina o màquina – màquina, la interfície gràfica assoleix el paper de l'àrbitre decidint el resultat en base als diversos esdeveniments que poden succeir: • Si un jugador abandona, guanya l'altre • Si els jugadors pacten, la partida es considera empat • Si un jugador rep escac i mat guanya l'altre jugador. • Si a sobre el taulell no hi queda material suficient per a que algun jugador pugui guanyar la partida es declara empat. • Si durant la partida es repeteix una mateixa posició tres cops i un dels jugadors demana taules la partida esdevé empat. • Si en els 50 últims moviments no s'ha fet cap captura de peça i no s'ha avançat cap peó la partida es declara empat. • Si el jugador al qual li toca moure es troba en situació d'ofegat[7] la partida es declara empat.			
Cursos alternatius:			
-			

Cas d'ús:	Oferir taules	**Actors:**	Jugador,Àrbitre
Propòsit:	Donar a cadascun dels jugadors el dret que té, després d'haver fet el seu moviment, a oferir taules al jugador contrari	**Tipus:**	Primari - Essencial

Resum:

Si un jugador estima que la posició actual no pot portar a la victòria a cap dels dos bàndols o bé no té ganes de continuar la lluita, pot oferir taules al rival immediatament després d'haver fet el seu moviment.

Requisits: F2,U7

Curs típic d'esdeveniments:

En el cas que el jugador actual estimi que la posició actual no pot portar a la victòria i/o no tingui ganes de continuar la partida, es procedirà de la següent manera:

1. El jugador que ha arribat a la conclusió de que la partida és taules realitza el seu moviment(evidentment, ha de ser un moviment que mantingui l'equilibri, que no el faci perdre).
2. Immediatament després d'haver mogut, ofereix taules al seu rival.
3. En aquest punt el seu rival té dues opcions
    a) Acceptar les taules: La partida s'acaba i l'àrbitre la declara empat.
    b) Refusar la oferta: La partida segueix amb normalitat.

Cursos alternatius:

-

Cas d'ús:	Acceptar taules	**Actors:**	Jugador,Àrbitre
Propòsit:	Que el jugador pugui acceptar la oferta del seu rival de taules	**Tipus:**	Primari - Essencial

Resum:

Un cop el seu rival ha mogut i li ha proposat taules, el jugador accepta la oferta i la partida acaba en empat.

Requisits: F2,U7

Curs típic d'esdeveniments:

Un cop el seu rival ha mogut i li ha proposat taules, el jugador accepta la oferta i la partida acaba en empat. L'àrbitre declara la partida com empat

Cursos alternatius:

-

Cas d'ús:	Abandonar	**Actors:**	Jugador
Propòsit:	Que un jugador que veu la seva derrota com inevitable pugui abandonar la partida.	**Tipus:**	Primari – Essencial

Resum:

Un dels jugadors arriba a la conclusió de que la partida està perduda i abandona.

Requisits: F2,U7

Curs típic d'esdeveniments:

2. El jugador candidat a abandonar calcula la posició actual i arriba a la conclusió de que, faci el que faci, el seu rival podrà trobar una seqüència d'accions que el porti a la derrota.
3. Davant la impotència que suposa la conclusió anterior, comunica al seu rival que abandona la partida.
4. L'àrbitre adjudica la partida al jugador que no ha abandonat.

Cursos alternatius:

-

Cas d'ús:	Jugar Partida Home - Màquina	**Actors:**	Jugador Humà, Jugador Artificial
Propòsit:	Portar a terme el cas d'ús UC1 en el cas particular de que la partida sigui home contra màquina	**Tipus:**	Primari – Real

Resum:

Partida, igual que la descrita al cas d'ús UC1, en el cas particular d'enfrontament Home – Màquina.

Requisits: F1,F2,F3,F5,U1,U2,U3,U4,U5,U7

Curs típic d'esdeveniments:

Actor 1: Jugador Blanques	Actor 2: Jugador Negres	Actor 3: Àrbitre
		1.L'àrbitre inicia la partida
Mentre la partida no s'hagi acabat		
2. Les blanques pensen quina és la millor jugada possible. a) Si el jugador és humà raona la jugada durant el temps que estimi oportú b) Si el jugador és artificial calcula la jugada durant el temps que estimi oportú mitjançant l'algorisme de cerca — **Realització 1**		
3. Un cop decidida la millor jugada, les blanques fan el moviment que creuen millor. a) Si el jugador és humà fa el moviment que creu que és millor mitjançant drag&drop b) Si el jugador és artificial fa el moviment comunicant-lo mitjançant el protocol UCI a la interfície gràfica que quan el rebi actualitzarà el tauler — **Realització 2**		
	Si no s'ha acabat la partida 4. Les negres pensen quina és la millor jugada possible. • Aplica el mateix que a la **Realització 1**	
	5. Un cop decidida la millor jugada, les negres fan el moviment que creuen millor. • Aplica el mateix que a la **Realització 2**	
		6. Si hi ha guanyador, l'àrbitre(**en aquest cas la interfície gràfica**) li adjudica la partida. En cas contrari, la declara taules

Cursos alternatius:

-

Cas d'ús:	Interacció Gràfica Home Màquina	**Actors:**	Jugador Humà, Interfície Gràfica
Propòsit:	Que el jugador humà pugui interactuar amb el sistema de forma gràfica	**Tipus:**	Primari – Real
Resum:			
La interfície gràfica tradueix els gestos fets per el jugador humà en missatges del protocol UCI			
Requisits: F2,U3,U4,U7			
Curs típic d'esdeveniments:			
1. El jugador humà realitza una acció a la interfície gràfica(i.e. moure una peça) 2. La interfície gràfica transforma l'acció de l'usuari en un missatge del protocol UCI i l'envia al jugador artificial			
Cursos alternatius:			
-			

Cas d'ús:	Comunicació protocol UCI	**Actors:**	Jugador Artificial, Interfície Gràfica
Propòsit:	Diàleg entre la interfície gràfica i el motor d'escacs	**Tipus:**	Primari – Real
Resum:			
La interfície gràfica tradueix els missatges que li arriben del jugador artificial al llenguatge gràfic de la interfície gràfica			
Requisits: F2,F5,U7			
Curs típic d'esdeveniments:			
1. El jugador Artificial envia un missatge del protocol UCI a la interfície reflectint algun canvi en el tauler. 2. La interfície gràfica actualitza la informació gràfica per tal de que el jugador humà pugui percebre el canvi.			
Cursos alternatius:			
-			

Cas d'ús:	Anàlisi partida	**Actors:**	Jugador Artificial, Jugador Humà
Propòsit:	Assistència del Jugador Artificial al Jugador Humà durant l'anàlisi d'una partida d'escacs	**Tipus:**	Primari – Real

Resum:

El jugador, durant la revisió/anàlisi de la partida en qüestió, va seleccionant diferents posicions que li són d'interès. Per a cada posició que es selecciona, la interfície gràfica treu per pantalla la puntuació de l'últim anàlisi al que s'hagi arribat per a la puntuació actual

Requisits: F1,F2,F4,F5,U7

Curs típic d'esdeveniments:

	Actor 1: Jugador Humà	Actor 2: Jugador Artificial	Actor 3: Interfície Gràfica
	1. Carregar Partida		
Mentre el jugador no finalitzi l'anàlisi	2.Selecció de Posició		
		3. Anàlisi infinit de la posició mentre no se'n seleccioni una altra.	3. Comunicació dels resultats obtinguts fins el moment al Jugador Humà

Cursos alternatius:

-

Cas d'ús:	Calcular Posició	**Actors:**	Jugador Artificial, Interfície Gràfica
Propòsit:	Diàleg entre la interfície gràfica i el motor d'escacs.	**Tipus:**	Primari – Real

Resum:

El jugador artificial calcula la posició que li indica la interfície gràfica mitjançant el protocol UCI

Requisits: F1,U6

Curs típic d'esdeveniments:

1. La interfície gràfica envia la posició que s'ha de calcular al Jugador Artificial mitjançant el protocol UCI.
2. El Jugador Artificial, mentre la interfície no li indiqui lo contrari calcula la posició que ha rebut i informa periòdicament a la interfície del resultat.
3. La interfície va presentant de manera gràfica els resultats parcials que rep del Jugador Artificial.
4. La interfície gràfica envia la ordre de parada al Jugador Artificial mitjançant el protocol UCI i aquest finalitza el procés de càlcul.

Cursos alternatius:

-

Cas d'ús:	Retrocedir Jugada	**Actors:**	Jugador Humà, Interfície Gràfica, Jugador Artificial
Propòsit:	Navegació dins la partida que s'analitza	**Tipus:**	Primari – Real

Resum:

El Jugador Humà retrocedeix una jugada per poder analitzar la posició que ve just abans de la posició actual

Requisits: U6

Curs típic d'esdeveniments:

1. El Jugador Humà decideix retrocedir una jugada i ho indica a la interfície gràfica
2. La interfície gràfica ho comunica al Jugador Artificial per protocol UCI que començarà a calcular la posició anterior.

Cursos alternatius:

-

Cas d'ús:	Avançar Jugada	**Actors:**	Jugador Humà,Interfície Gràfica, Jugador Artificial
Propòsit:	Navegació dins la partida que s'analitza	**Tipus:**	Primari – Real

Resum:

El Jugador Humà avança una jugada per poder analitzar la posició que ve just després de la posició actual

Requisits: U6

Curs típic d'esdeveniments:

1. El Jugador Humà decideix avançar una jugada i ho indica a la interfície gràfica
2. La interfície gràfica ho comunica al Jugador Artificial per protocol UCI que començarà a calcular la posició següent.

Cursos alternatius:

-

Cas d'ús:	Ramificar Posició	**Actors:**	Jugador Humà,Interfície Gràfica, Jugador Artificial
Propòsit:	Avaluar el guany/pèrdua de fer una jugada diferent a la que s'ha fet	**Tipus:**	Primari – Real

Resum:

El Jugador Humà crea una nova branca/variant que serà analitzada per el Jugador Artificial.

Requisits: U6

Curs típic d'esdeveniments:

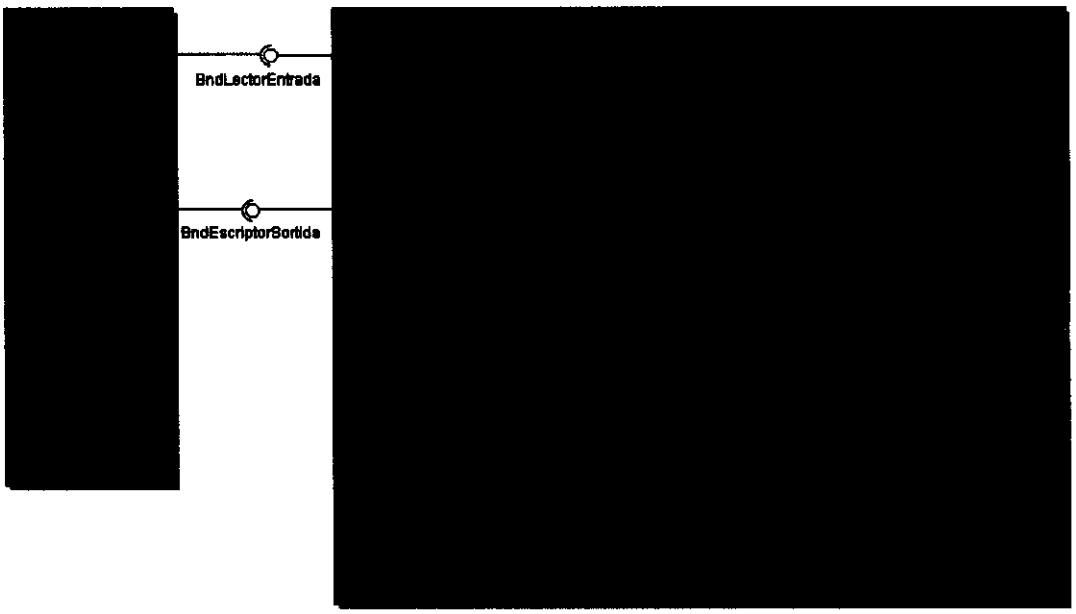
1. Per crear la nova variant, el Jugador Humà introdueix una jugada diferent a totes les jugades següents existents.
2. La interfície gràfica crea la nova variant i es posiciona en aquesta última posició comunicant-ho al Jugador Artificial que començarà a calcular la posició.

Cursos alternatius:

-

4.2 Descomposició del sistema en mòduls funcionals

Després d'haver vist a l'apartat anterior totes les funcionalitats que ha de suportar el sistema, en aquest apartat farem una divisió d'aquestes en diversos mòduls funcionals que ens guiarà després durant el disseny tècnic a l'hora de classificar les classes en diversos diagrames i paquets. A continuació presentem un diagrama de components d'alt nivell on s'especifica quins serveis oferirà cada mòdul funcional i quins serveis consumirà.



4.2.1 Interfície Gràfica

Aquest component s'encarrega de totes les funcionalitats relacionades amb la interacció amb l'usuari. Simbolitza qualsevol software que es vulgui utilitzar que implementi el protocol UCI(Universal Chess Interface). S'inclou en el diagrama de cara a situar millor el motor d'intel·ligència en el seu context encara que no s'ha d'implementar res al respecte ja que, com s'ha comentat amb anterioritat, la implementació de la interfície gràfica queda fora de l'abast d'aquest projecte.

4.2.2 Motor d'intel·ligència

És el component que recull tota aquella funcionalitat relacionada amb jugar a escacs de manera intel·ligent i analitzar partides. Vist des de fora, el que fa aquest component és llegir els missatges UCI que li envia la interfície gràfica, fer les accions que se li encomanen i retornar els resultats obtinguts via missatges UCI un altre cop cap a la interfície gràfica. Donada la seva gran complexitat s'ha dividit en 6 submòduls que es detallen a continuació.

4.2.2.1 Controlador Motor

És l'encarregat d'interpretar tots els missatges UCI que rep el motor i orquestrar els altres components per tal de que facin la feina que la interfície gràfica ha demanat. Un cop la feina s'ha acabat transforma els resultats en nous missatges UCI que envia a la interfície gràfica.

Controlador Motor: Serveis Oferts		
Lectura Entrada	La seva missió principal és la de capturar els missatges UCI que arriben des de la interfície gràfica.	
Escriptura Sortida	Un cop acabada la feina sol·licitada per la interfície gràfica, transforma els seus resultats en missatges UCI i els hi envia per tal de donar-li feedback.	
Serveis consumits		
Configuració Posició	Representació de posicions	Un cop es fixa la posició a la interfície gràfica, aquesta la transmet al controlador. El controlador un cop la rep l'anivella amb la posició del motor via aquest servei.
Configuració Cerca	Algorísmica de Cerca	Abans d'enviar l'ordre que desencadenarà tot el procés de cerca, el controlador pot tenir la necessitat de parametritzar algun aspecte d'aquest procés.
Control Cerca		El Controlador és l'encarregat de, a petició de la interfície gràfica, iniciar i parar el procés de cerca
Monitorització Cerca		Mentre el mòdul de cerca està buscant la millor jugada que es pot fer en una posició determinada, el Controlador del Motor té la responsabilitat de monitoritzar aquest procés i enviar informació periòdica a la interfície gràfica. En concret la informació que més interessa és: número de nodes buscats, velocitat de la cerca en nodes/segon i la variant principal trobada fins el moment actual.

4.2.2.2 Representació de posicions

La responsabilitat d'aquest mòdul és la d'oferir a la resta totes les funcionalitat relacionades amb la representació fidedigne de posicions d'escacs.

Representació de Posicions: Serveis Oferts	
Nom del servei	**Descripció del servei**
Configuració posició	Aquesta servei permet configurar la posició actual a partir d'una posició en format FEN
Lectura/Escriptura posició	Permet poder consultar en qualsevol moment tots els detalls de la posició actual així com aplicar i desfer moviments a la posició actual.
Serveis consumits: N/A	

4.2.2.3 Algorísmica de Cerca

Aquest mòdul engloba tota la funcionalitat relacionada amb la recerca de la millor jugada disponible des de la posició actual.

Algorismica de Cerca: Serveis Oferts		
Nom del servei	**Descripció del servei**	
Control Cerca	Aquesta funcionalitat permet, bàsicament, engegar i parar el procés de cerca	
Configuració Cerca	Permet, abans d'engegar el procés de cerca, configurar-ne els seus paràmetres(i. e. Profunditat màxima, temps màxim de cerca, etc..)	
Monitorització Cerca	Permet conèixer, en temps real, l'estat de la cerca que s'està executant actualment. Gràcies a aquesta funcionalitat es podrà saber en tot moment quants nodes s'han buscat, quines és la velocitat en nodes/segon del motor de cerca, quina és la variant principal[5] actual, etc...	
Serveis consumits		
Nom del servei	**C. Servidor**	**Finalitat del consum**
Generació de Moviments	Generació Moviments	Per tal de poder fer l'expansió de l'arbre de cerca es necessita saber, donada una posició, quines són els moviments disponibles en aquesta per tal de poder generar les posicions successores.
Aplicar Moviment		Un cop es sap quines són els moviments disponibles en una posició donada, l'algorisme de cerca ha de poder provar-les una darrera l'altre sistemàticament.
Desfer Moviment		Quan ja s'ha calculat el valor d'una jugada, aquesta s'ha de poder desfer per calcular el valor de les altres jugades disponibles en la mateixa posició.
Avaluació de Posicions	Avaluació Posicions	Quan s'arriba a posicions terminals, s'ha d'avaluar el valor que tenen aquestes per tal de poder pujar-les després cap als nivells superiors segons la dinàmica de l'algorisme Minimax/Alphabeta

4.2.2.4 Generació de Moviments

Engloba tota la lògica referent a, donada la posició actual, generar totes les posicions possibles que la succeeixen.

Generació de Moviments: Serveis Oferts		
Nom del servei	**Descripció del servei**	
Generació de moviments	Donada la posició actual, genera totes les jugades possibles que es poden fer des de la mateixa.	
Aplicar moviment	Permet aplicar la jugada *i* a la posició actual	
Desfer moviment	Permet desfer l'últim moviment que s'ha aplicat al tauler.	
Serveis consumits		
Nom del servei	**C. Servidor**	**Finalitat del consum**
Lectura/Escriptura posició	Representació de posicions	Per tal de poder generar els moviments disponibles en la posició actual, el sistema ha de poder llegir els detalls de la posició en que ens trobem. Per altra banda, per poder aplicar i desfer moviments, necessita poder escriure i fer canvis en la posició.

4.2.2.5 Avaluació de Posicions

Gràcies a aquest mòdul, l'algorisme de cerca pot puntuar i comparar posicions i així escollir la jugada que plausiblement el portarà a la millor possible. Aquesta puntuació es fa via una sèrie de coneixements heurístics que es coneixen gràcies a l'experiència i coneixement generat a partir de milions de partides entre els millors mestres d'escacs.

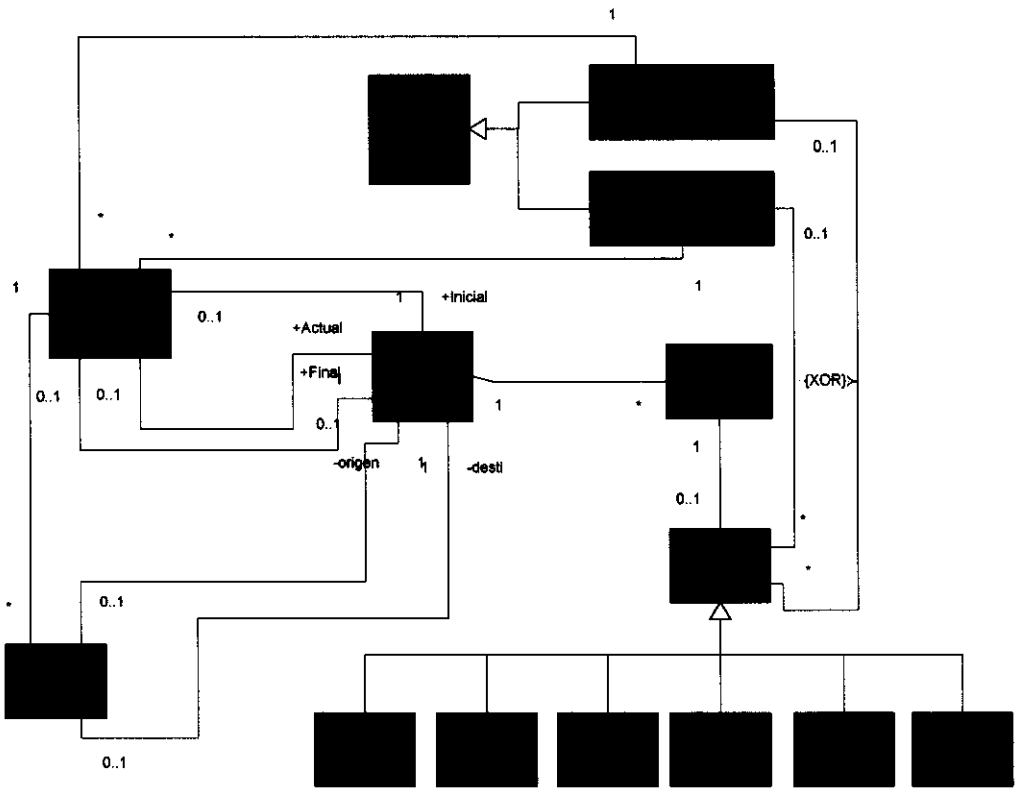
Avaluació de Posicions: Serveis Oferts		
Nom del servei	**Descripció del servei**	
Avaluació de la posició	Aquest servei possibilita avaluar la posició actual. Engloba tot el coneixement heurístic que guia a la cerca.	
Serveis consumits		
Nom del servei	**C. Servidor**	**Finalitat del consum**
Lectura/Escriptura posició	Representació de posicions	Per tal de poder avaluar el valor de la posició actual, el sistema ha de poder llegir els detalls de la posició en que ens trobem.

4.2.2.6 Gestió de Components

Aquest mòdul és transversal als altres que s'encarrega de gestionar la creació i interconnexió de cadascun dels altres components. La seva descripció, per tant, es farà a l'apartat de disseny tècnic.

4.3 Model conceptual del domini

En el següent diagrama s'il·lustren els conceptes principals del domini d'aplicació del sistema així com també la interrelació entre ells. Després del diagrama es presenta un llistat dels mateixos adjuntant al costat una breu descripció de cadascun d'ells.



Principals entitats	
Partida	Conjunt de jugades que es fan a partir de la posició inicial del joc i que desencadena amb el resultat de taules, victòria blanca o victòria negra.
Jugada	Acció que fa el jugador que té el torn movent una peça d'una casella d'origen a una casella de destí amb la possibilitat de capturar o no una peça de l'equip rival
Posició	Situació en un moment puntual de la partida que es composa de la disposició particular de les peces i del jugador al qual li toca moure
Casella	Unitat mínima en la que es divideix un tauler d'escacs
Peça	Figura d'un determinat color i tipus de la que disposa un jugador per fer diferents jugades i captures de peces rivals.
Peó	Tipus de peça
Alfil	Tipus de peça
Cavall	Tipus de peça
Torre	Tipus de peça
Dama	Tipus de peça
Rei	Tipus de peça
Jugador	Rol que agafa la persona(o sistema) que juga a escacs segons el qual se li assignen les peces d'un determinat color i la missió de capturar al rei del color contrari
Jugador peces blanques	Jugador al qual se li assignen les peces de color blanc i amb elles l'avantatge de fer el primer moviment
Jugador peces negres	Jugador al qual se li assignen les peces de color negre.

5. Disseny tècnic del sistema

5.1 Eines de desenvolupament

En aquest apartat s'enumeren totes aquelles eines que s'han utilitzat durant el desenvolupament tècnic del producte. El públic que utilitza sistemes com el que ens ocupa és limitat lo que provoca que els beneficis siguin reduïts i que per tant s'hagin d'optimitzar, com es veurà a l'apartat d'anàlisi econòmica, les despeses al màxim. És per això que una de les premisses ha estat pagar el mínim de llicències el que ha provocat que el projecte s'ha desenvolupat gairebé íntegrament amb Open Source. Amb això no volem dir que la única bondat del software que hem utilitzat és que sigui gratuït, res més allunyat de la realitat, el software que hem utilitzat ha demostrat ser robust, versàtil, funcional, amb suport de la comunitat i amb el valor afegit de poder-lo modificar i adaptar-lo a les nostres necessitats.

Software	Descripció
BOUML	**Software de modelat UML.** S'ha utilitzat per fer l'anàlisi i disseny tècnic orientat a objectes i el diagrames d'especificació. Un dels seus punts forts, a part de que la interfície de disseny està ben aconseguida, és que genera el codi esquelet de les classes en C++ per després acabar de completar-lo a mà.
MinGW GCC	**Compilador del projecte GNU.** És un compilador que permet programar en diversos llenguatges. Per el cas del projecte s'ha utilitzat només el compilador de C++. El codi màquina que genera és força bo i existeixen implementacions per a la majoria de plataformes el que possibilita la portabilitat del codi.
eclipse ECLIPSE	**Entorn integrat de desenvolupament(IDE).** És un dels entorns de desenvolupaments lliures més potents disponible a la xarxa. El seu sistema de plugins permet que els desenvolupadors puguin anar ampliant les seves funcionalitats. Per el cas de desenvolupar en C++ disposa d'un plugin que facilita les tasques del programador amb prestacions com debugat integrat o resaltat de sintaxi.
GDB	**Debugador del projecte GNU.** L'utilització d'aquest software ha permès debugar els errors que anaven sorgint durant el desenvolupament amb rapidesa i eficiència.
Qt Code less. Create more. Deploy everywhere. Llibreria QT	**Llibreria de components C++ multiplataforma.** Llibreria gràfica desenvolupada per l'empresa Trolltech recentment comprada per Nokia. Coneguda arreu del mon per les seves prestacions, robustesa i portabilitat entre plataformes. Per el cas d'aquesta projecte s'ha utilitzat la llibreria de threads a causa de que els requisits funcionals(en concret la implementació del protocol UCI) impedien que el programa pogués executar-se amb un sol fil/procés.
boost Llibreria BOOST	**Llibreria de C++.** Llibreria de prestigi en el desenvolupament de la qual hi participen experts de prestigi que també estan participant en la definició del nou estàndard de C++ conegut com C++0x. S'ha utilitzat una sublibreria que soluciona molt bé tots els temes relacionats amb proves unitàries.
ARENA	**Interfície gràfica per a motors d'escacs.** Implementa el protocol UCI i permet funcionalitats avançades com per exemple torejos entre motors. En aquest cas cal matisar que el software **no és Open Source encara que sí és gratuït.**

5.2 Principals directrius i patrons de disseny

5.2.1 Orientació a objectes vers eficiència

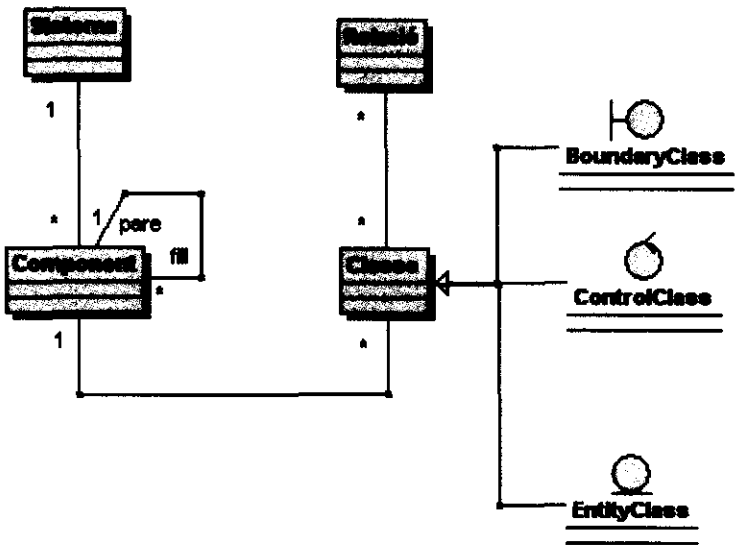
La majoria d'implementacions de motors d'escacs mínimament competitives existents estan escrites en C i les que ho estan en C++ acostumen a utilitzar un dialecte de C++ que no utilitza les prestacions d'orientació a objectes. Amb aquest panorama, la decisió de dissenyar o no el sistema amb orientació a objectes va ser difícil. Al final però es va decidir que sí ja que es va creure que si es tenia prou cura en algunes parts crítiques del codi, la penalització en eficiència no havia de ser tan gran i que es guanyaria molt en mantenibilitat i extensibilitat del software. No obstant, per a que això fos possible s'havien de definir unes normes de desenvolupament que prohibissin aquelles pràctiques que més podien perjudicar el rendiment, les quals es detallen en el següent apartat. Amb tot, els resultats no s'allunyen molt del que s'havia previst, el codi és força eficient (el motor és capaç d'analitzar unes 300.000 posicions per segon) i el codi s'entén millor que el d'altres implementacions que estan escrites en C.

5.2.2 Criteris per utilitzar la orientació a objectes de manera eficient

#	Norma	Justificació
N1	Pas de paràmetres d'objectes pesats per referència, mai per valor	Això evitarà que rutines crítiques del codi, com la de cerca, inverteixin massa de temps fent còpies. Ex: En la funció que avalua posicions, que s'executa milions de vegades seria molt costós que cada funció que s'avalua es copiés a la pila.
N2	No instanciar objectes en parts crítiques del codi. Utilitzar en el seu lloc pools d'objectes creats durant la inicialització del sistema	La creació d'objectes és costosa ja que suposa fer crides al sistema operatiu per reservar memòria lo qual no sempre és trivial. En lloc de que la memòria es reservi en zones crítiques del codi el que es fa és crear un sol cop al inici del programa pools d'objectes(per exemple moviments) que es poden anar reutilitzant. Amb això s'aconsegueix un gran estalvi en temps.
N3	Fer un ús moderat de les funcions virtuals i evitar-ne l'ús a les zones més crítiques del codi	El fet de que una funció sigui virtual suposa un decrement petit en el rendiment però de gran importància si la funció es cridada milions de vegades. Evitar-ne el seu ús.

5.2.3 Patró general de disseny dels mòduls funcionals del sistema.

En l'apartat 4.2 hem descompost la funcionalitat del sistema en diversos mòduls funcionals. Ara el tema que ens ocupa és el d'elaborar un esquema general que ens guiï a l'hora de mapejar aquests mòduls funcionals en classes de disseny. Anomenarem a aquest esquema el *metamodel* del sistema. El següent esquema il·lustra el model que ens ha permès dissenyar tot el sistema de manera estandarditzada.

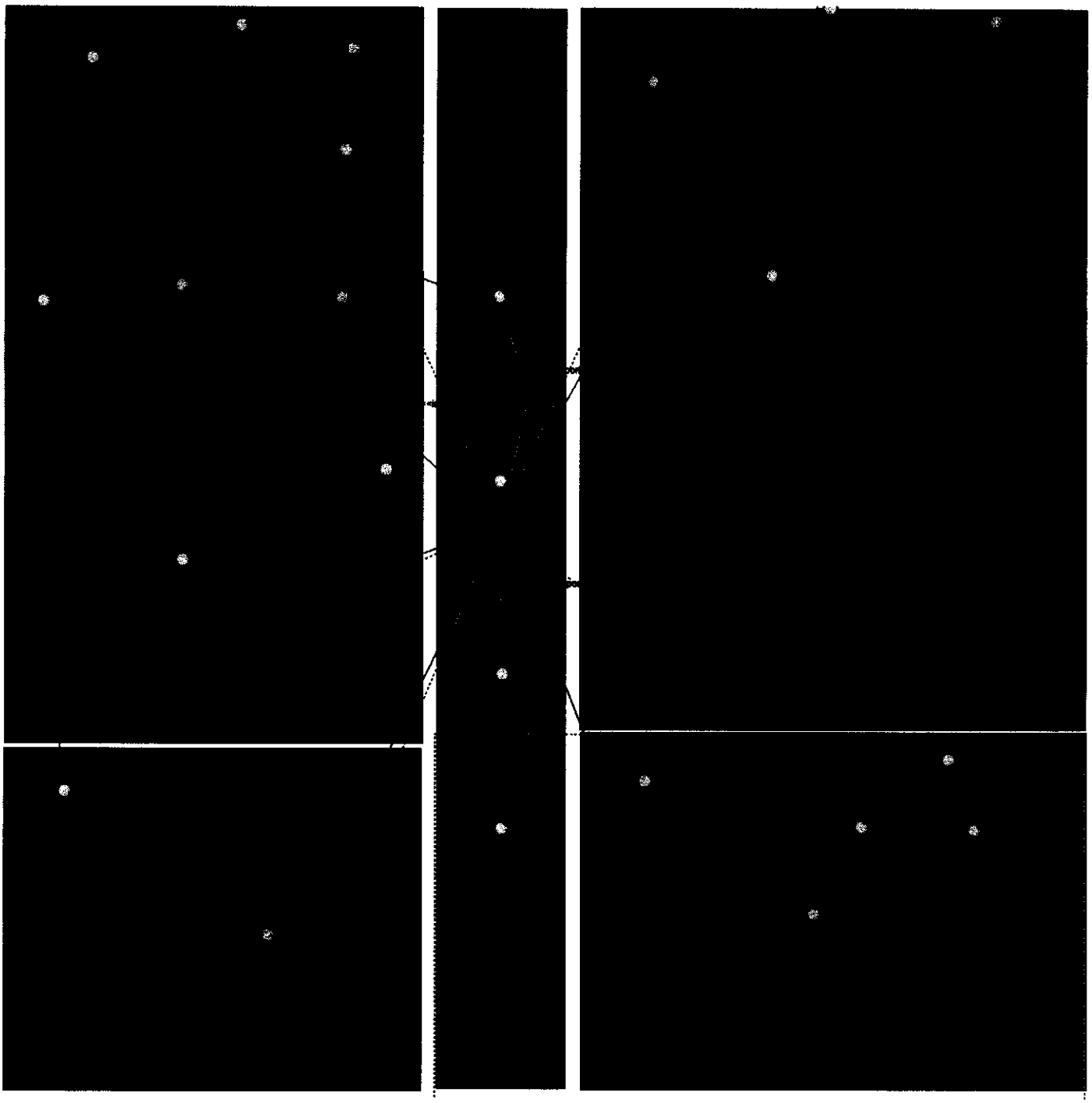


A continuació descriurem la funció de cada entitat del diagrama anterior així com també la seva materialització física:

Metamodel del sistema		
Sistema	Representa en el nostre cas el global del motor d'intel·ligència	La seva materialització física és un fitxer executable que es connecta amb una interfície gràfica que implementa el protocol UCI
Component	Representa cadascun del mòduls funcionals discutits a l'apartat 4.2	Classe C++ el nom de la qual comença amb el prefix Cmp. Aquesta classe normalment s'implementarà amb el patró d'enginyeria del software *Singleton* i la seva instància contindrà diversos apuntadors a instàncies de classes *Boundary,Control* i *Entity.*
Classe	Representa cadascuna de les classes que no són classes de Component	Es materialitzen via classes de C++. La nomenclatura es pot veure a cadascun dels subtipus pertinents.
BoundaryClass	Representa classes que fan la funció d'interfície d'un mòdul funcional. Tota la funcionalitat d'un mòdul que es consumeixi des d'altres mòduls s'ha de consumir des dels mètodes exposats per aquestes classes	Es materialitza via classes de C++ el nom de les quals comença per Bnd.
ControlClass	Representa classes que fan la funció de controlador del fluxe d'execució de la lògica d'un mòdul.	Es materialitza via classes de C++ el nom de les quals comença per Cnt.
EntityClass	Representa a classes que representen conceptes del domini de l'aplicació com pot ser per exemple una posició donada en un tauler d'escacs	Es materialitza via classes de C++ el nom de les quals comença per E.
Relació	Representa a relacions entre classes de diferents tipus	En la majoria dels casos es materialitzarà mitjançant atributs de classes C++. En algun cas especial seria possible representar-ho com una classe(i.e associacions amb cardinalitat N-N) encara que en aquest projecte no ha estat necessari.

5.3 Disseny tècnic dels components del sistema.

En el següent diagrama es pot veure la descomposició que s'ha fet del sistema en components d'acord amb els mòduls funcionals identificats a l'apartat 4.2.



En la part central del diagrama en blau es poden veure els components principals del programa que l'únic que fan és aglutinar els diferents subcomponents que aporten la funcionalitat del sistema. Com s'ha indicat a l'anterior apartat, totes les classes que comencen en Cmp representen components i les que comences per Bnd són classes boundary(frontera) que exposen la funcionalitat del mòdul als altres mòduls. Com es pot observar, per motius de

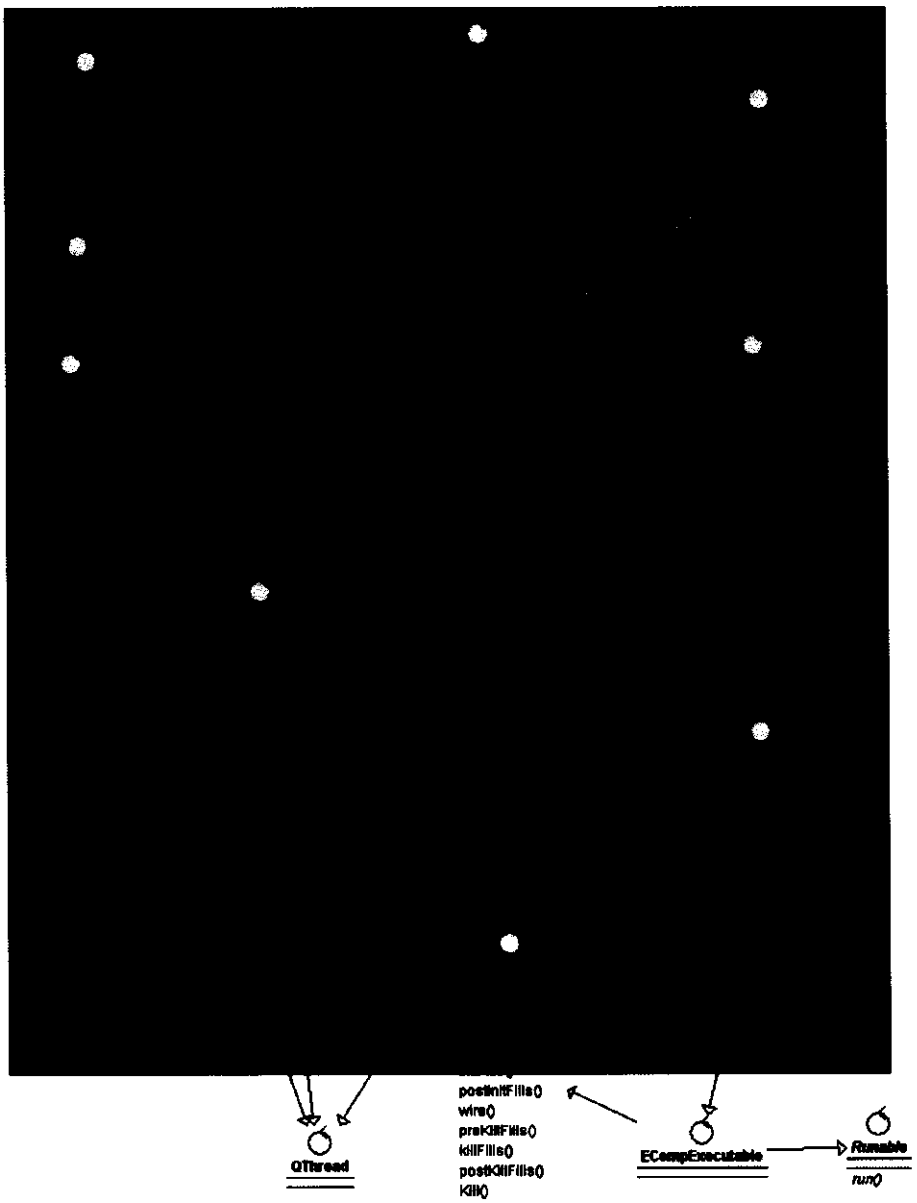
claredat, en aquest primer diagrama només s'han dibuixat les classes Boundary(interfícies dels components), els components i els subcomponents. En els següents subapartats descriurem amb més detall cadascun dels components, les classes que el composen i els fluxes d'execució més importants associats a cadascun d'ells.

5.3.1 Disseny del component Controlador del Motor

5.3.1.1 Estructura interna

En el següent diagrama es presenta de manera més detallada l'estructura interna del component *Controlador Motor* les funcions principals del qual són les següents:

- Comunicació amb la interfície gràfica mitjançant el protocol UCI.
- Orquestració i coordinació dels serveis oferts per els altres mòduls per tal d'acabar enviant a la interfície gràfica la informació que sol·licita.



5.3.1.2 Classe CmpControladorMotor

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

A continuació presentem una sèrie de diagrames de seqüència que il·lustren gràficament el comportament associat als mètodes més importants d'aquesta classe.

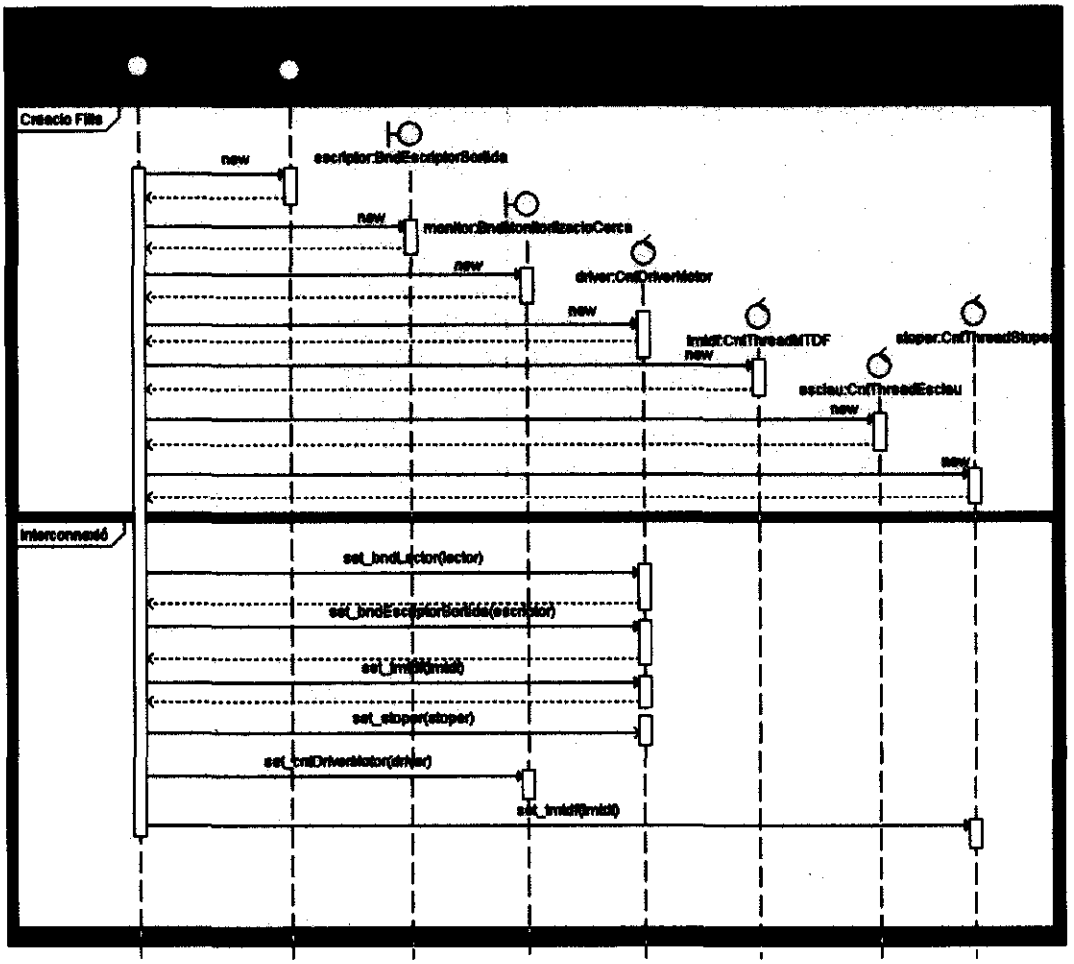


Diagrama S1

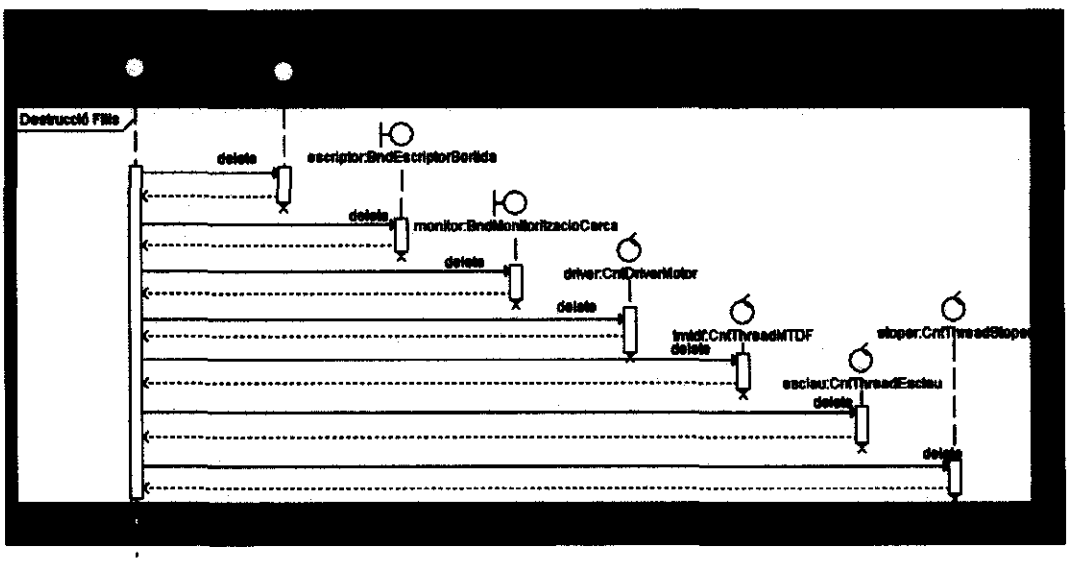


Diagrama S2

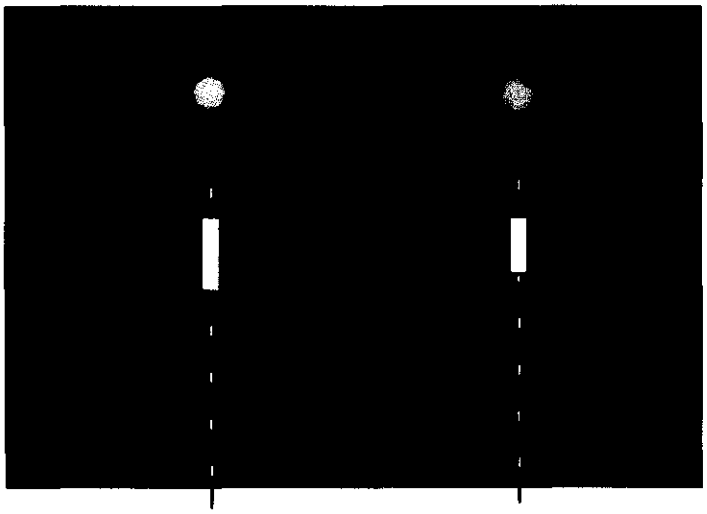
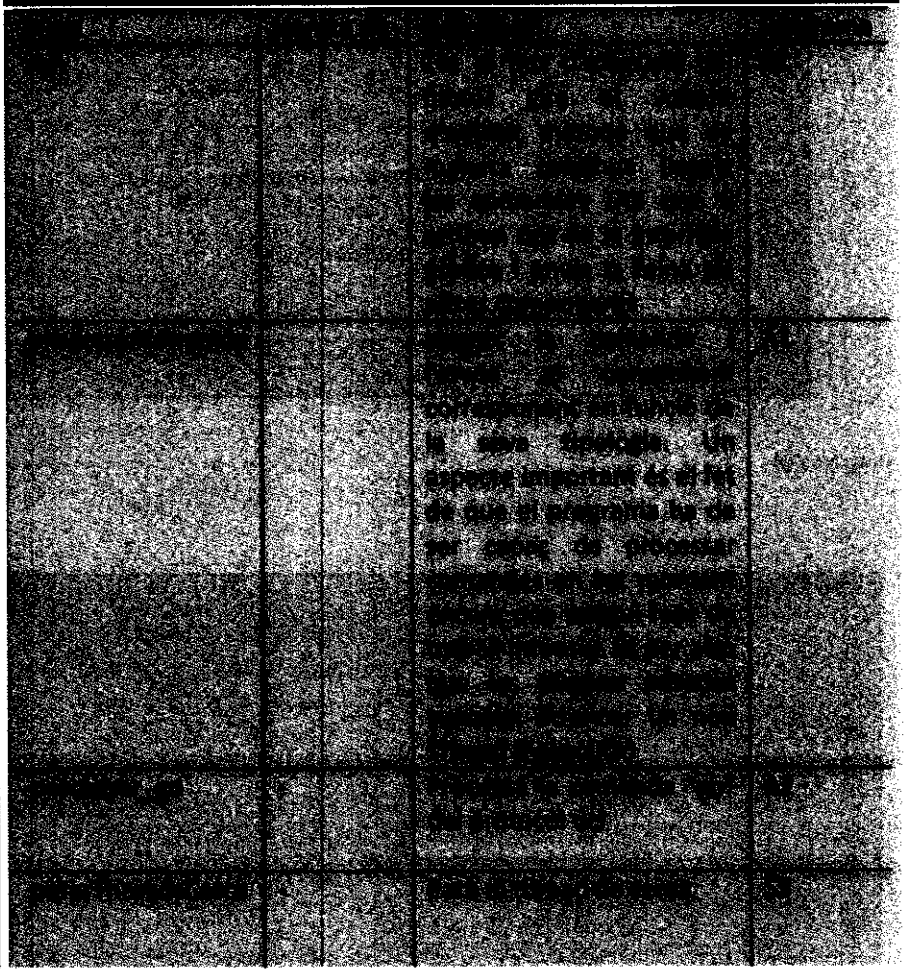


Diagrama S3

5.3.1.3 Classe CntDriverMotor

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

A continuació presentem una sèrie de diagrames de seqüència que il·lustren gràficament el comportament associat als mètodes més importants d'aquesta classe.

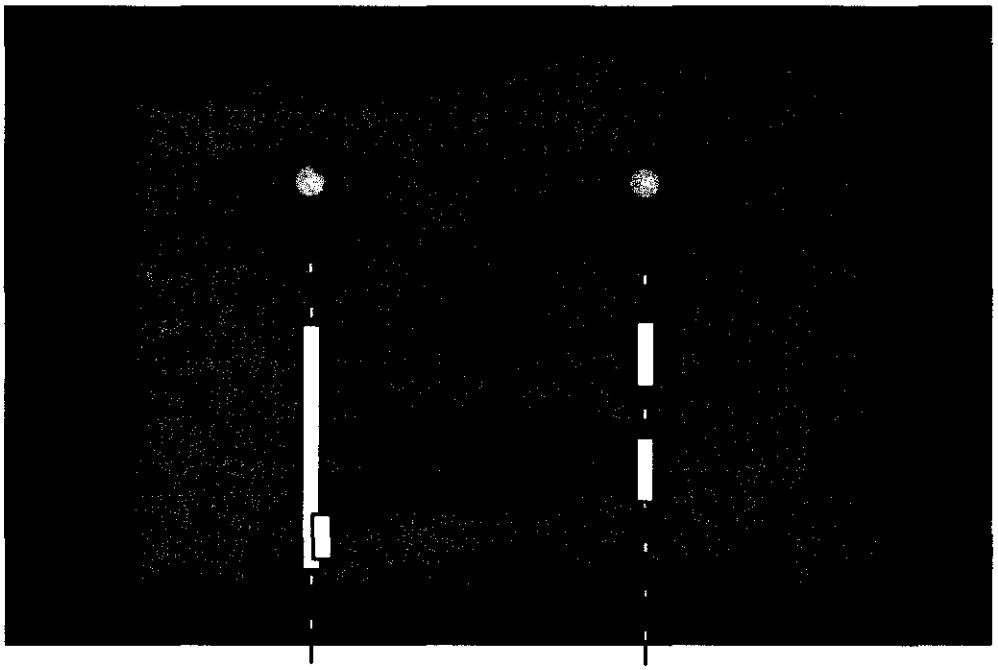


Diagrama S4

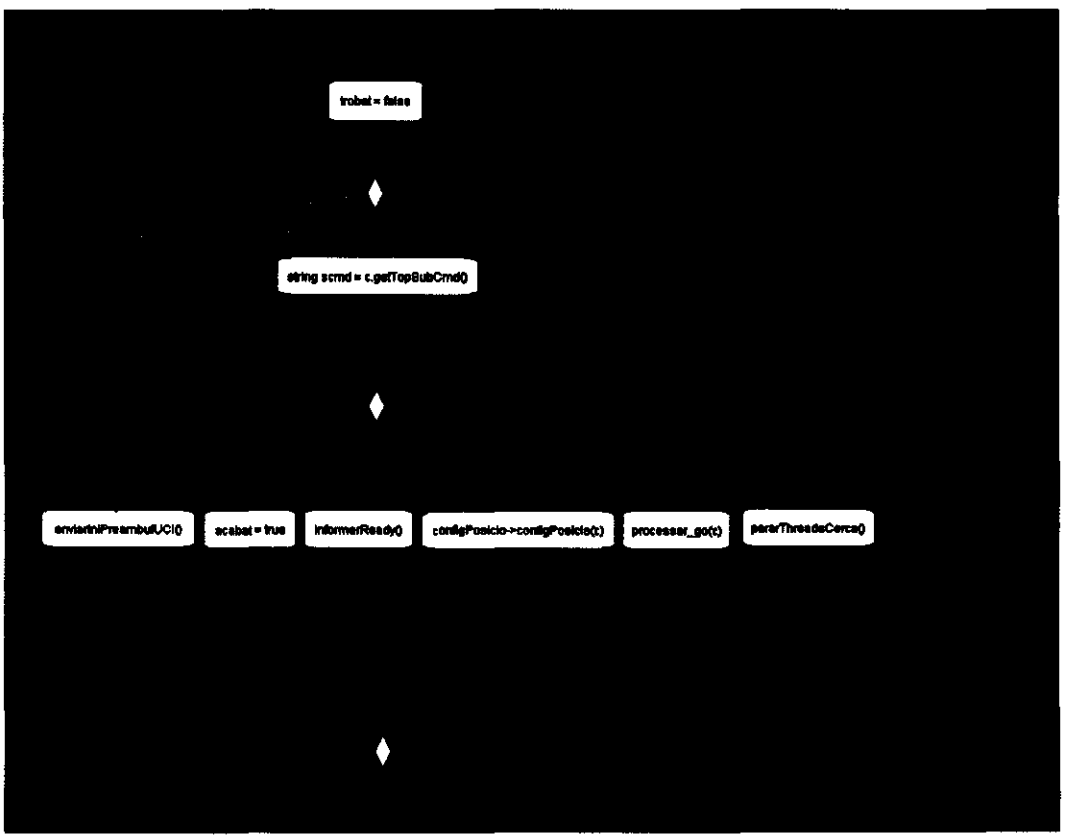


Diagrama A1

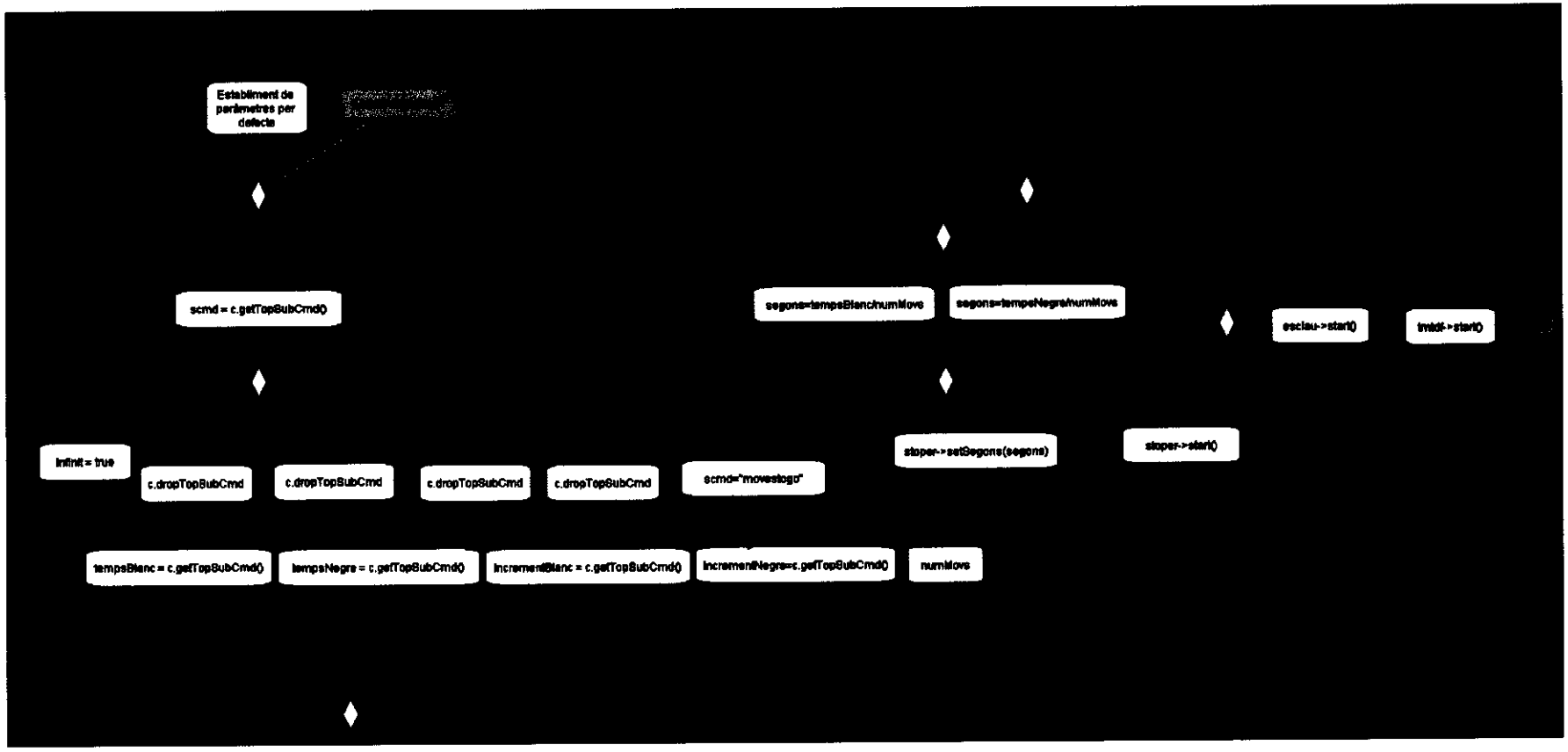

Establiment de paràmetres per defecte
scmd = c.getTopSubCmd()
segons=tempsBlanc/numMove
segons=tempsNegre/numMove
esclau->start()
mstr->start()
infinit = true
c.dropTopSubCmd
c.dropTopSubCmd
c.dropTopSubCmd
c.dropTopSubCmd
scmd="movestogo"
stoper->setSegons(segons)
stoper->start()
tempsBlanc = c.getTopSubCmd()
tempsNegre = c.getTopSubCmd()
incrementBlanc = c.getTopSubCmd()
incrementNegre=c.getTopSubCmd()
numMove

Diagrama A2

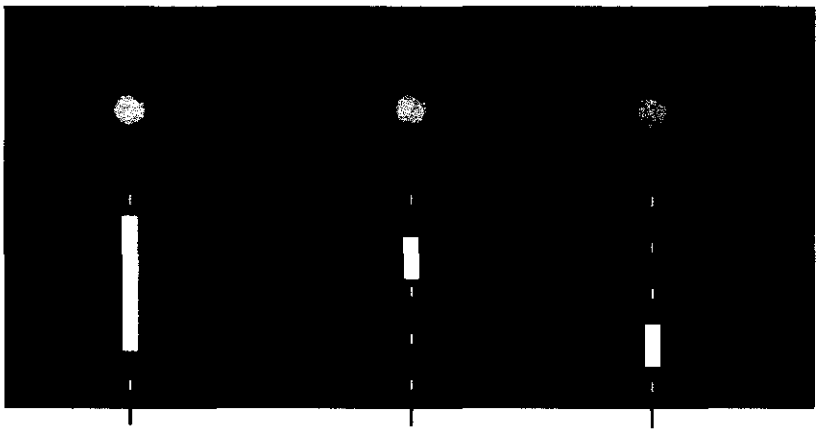
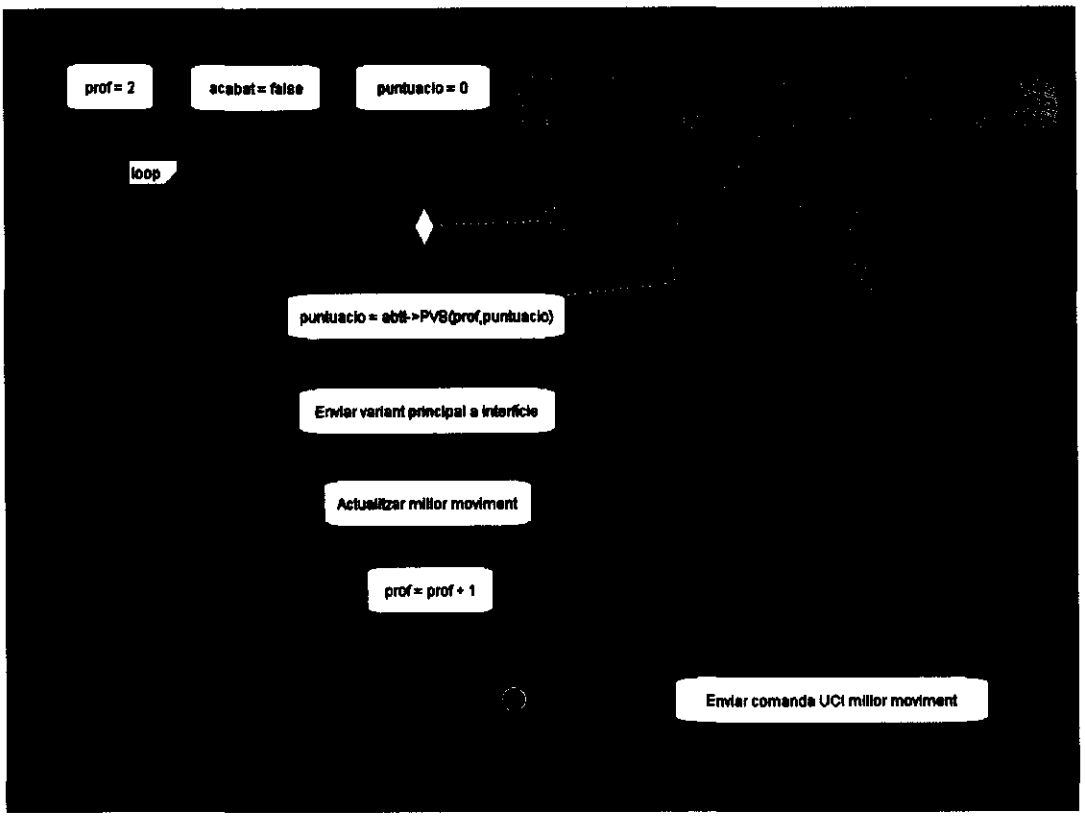


Diagrama S5

5.3.1.4 Classe CntThreadMTDF

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

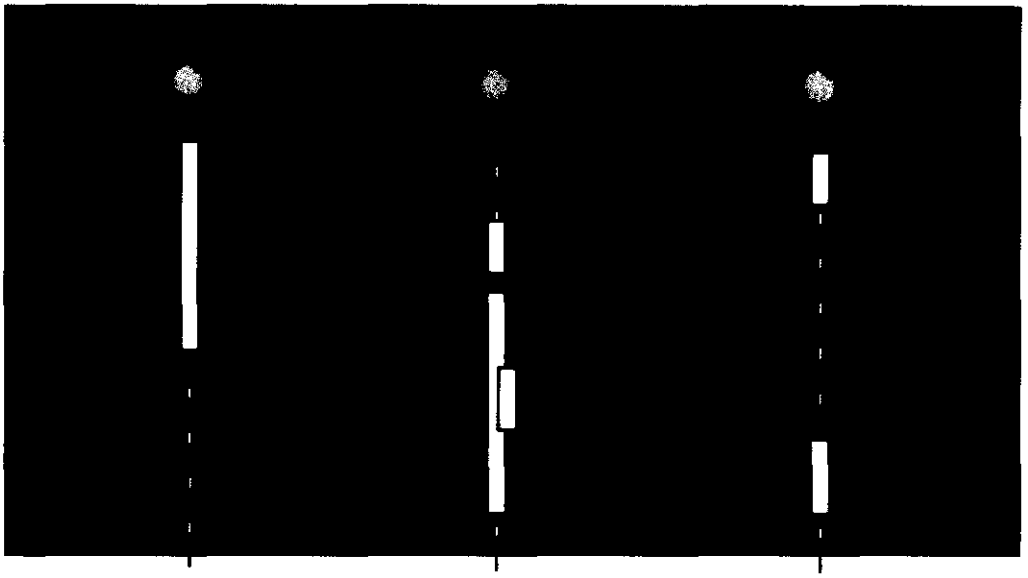
A continuació presentem una sèrie de diagrames de seqüència que il·lustren gràficament el comportament associat als mètodes més importants d'aquesta classe.



5.3.1.5 Classe CntThreadStoper

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

A continuació presentem un diagrama de seqüència que il·lustra gràficament el comportament associat als mètodes més importants d'aquesta classe.



5.3.1.6 Classe EComandaUCI

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.1.7 Classe BndLectorEntrada

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.1.8 Classe BndEscriptorSortida

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.1.9 Classe CntThreadEsclau

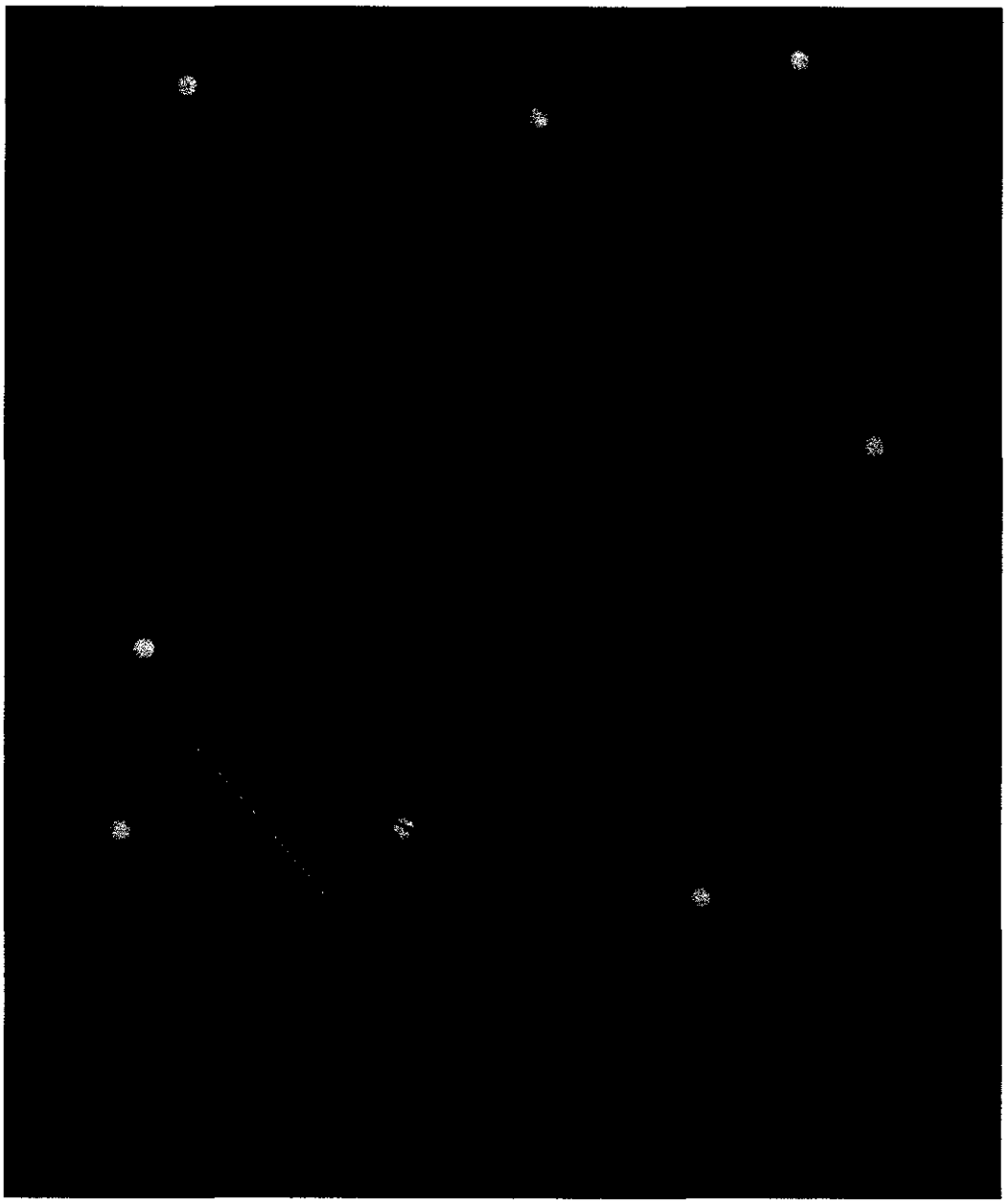
Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.2 Disseny del component Representació de Posicions

5.3.2.1 Estructura interna

En el següent diagrama es presenta de manera més detallada l'estructura interna del component *Representació Posicions* les funcions principals del qual són les següents:

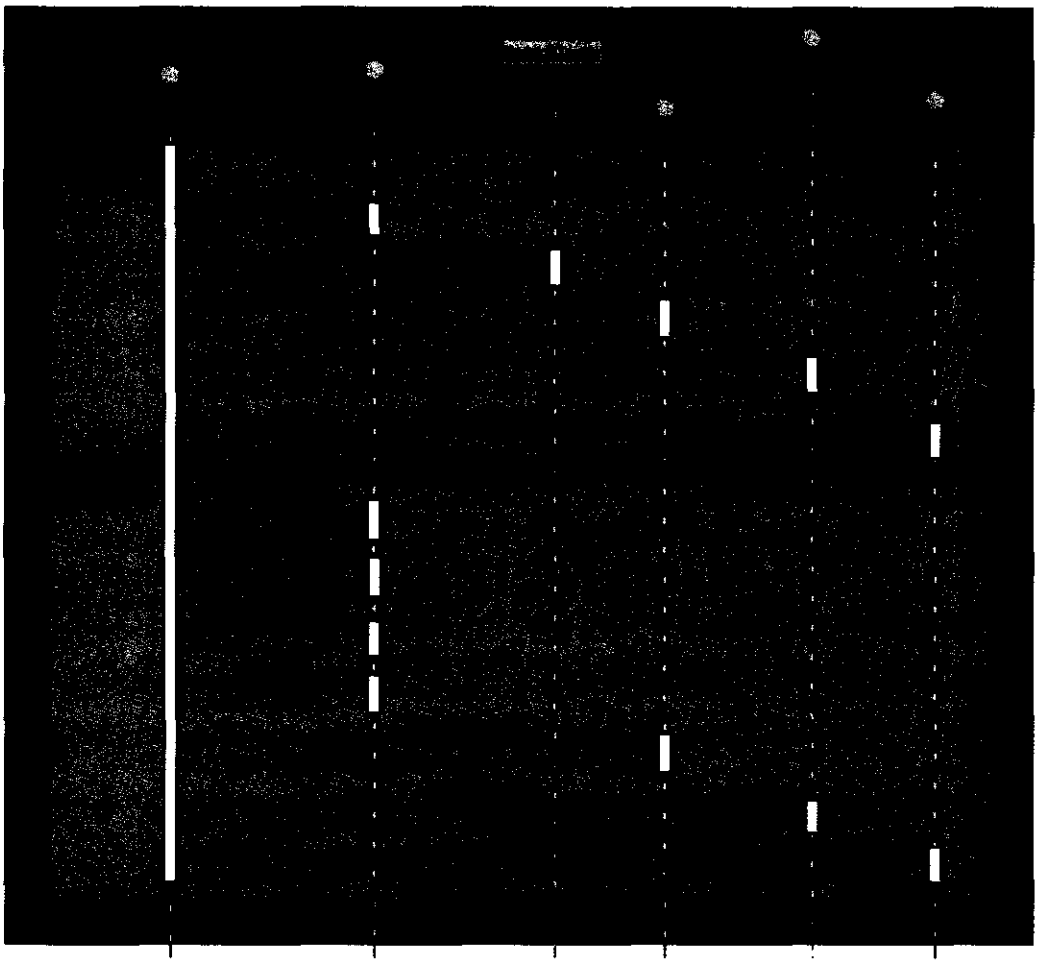
- Representació de la posició actual del tauler
- Parsing de posicions en format FEN i traducció al model de dades intern.

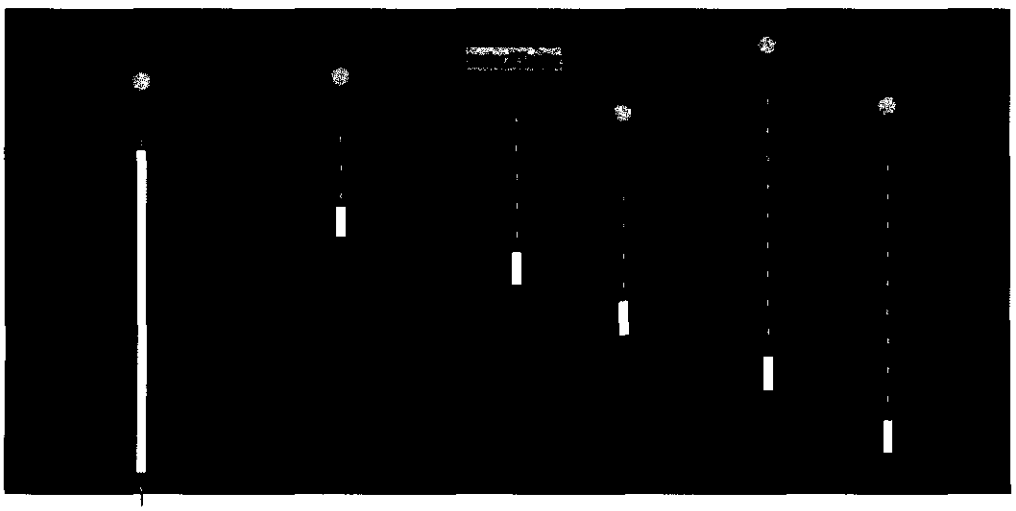


5.3.2.2 Classe CmpRepPosicions

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

A continuació presentem un diagrama de seqüència que il·lustra gràficament el comportament associat als mètodes més importants d'aquesta classe.





5.3.2.3 Classe ETauler

Dades generals	
Nom:	ETauler
Tipus:	Entitat
Descripció:	Representa l'estat actual en el que es troba el tauler(posició). Donat que és una estructura de dades que s'actualitza contínuament durant el procés de cerca s'han deixat els seus atributs com públics per evitar l'overhead que suposa accedir a ells via operacions.

Operacions més importants:

Nom	Hereta de	Descripció	Diagrama
copy	-	El tauler en el qual s'invoca el mètode esdevé una còpia exacte del tauler que se li passa com a paràmetre	-

Atributs més importants

Nom	Descripció
BitBoards	Vector de BitBoards que representa la situació de cadascuna de les peces de cadascun dels colors segons l'esquema descrit a l'apartat 2.2. A part de contenir un BitBoard per cada tipus de peça i color en conté dos de derivats(un que conté el conjunt de totes les peces blanques i l'altre el conjunt de peces negres) que faciliten i optimitzen alguns càlculs.
EstatEnroc	Vector que conté per a cada color dues entrades que indiquen si pot enrocar o no per cadascun dels dos flancs del tauler(de rei o de dama).
NumMitjosMovs50MR	Comptatge del número de moviments a efectes d'unes possibles taules a causa de la regla dels 50 moviments.
NumMovs	Número de moviments(no mitjos moviments) que s'han fet durant la partida
JugadorActual	Color que té el torn de jugar
ValorMaterial	Indica el valor material resultant de sumar la puntuació de les peces blanques i restar el de les negres
InfoEscac	Flag que indica si el rei del bàndol al qual li toca moure és troba en escac o no
NumPeces	Vector amb el número de peces per tipus i color
ValorPC	Indica per a cada tipus i color la suma del valor de les caselles que ocupen les peces
Key	1ª clau de hash calculada amb l'algorisme de Zobrist
KeyLock	2ª clau (per detectar col·lisions) calculada amb l'algorisme de Zobrist

5.3.2.4 Classe BndConfiguracioPosicio

Dades generals			
Nom:	BndConfiguracioPosicio		
Tipus:	Boundary		
Descripció:	La seva missió és la de configurar la posició a partir de les comandes UCI rebudes de la interfície gràfica		
Operacions més importants:	**Nom**	**Hereta de**	**Descripció**	**Diagrama**
	configPosició	-	Estableix tots els atributs del tauler a partir de la comanda UCI rebuda com a paràmetre la qual és la posició en format FEN(Forsyth-Edwards Notation)	-

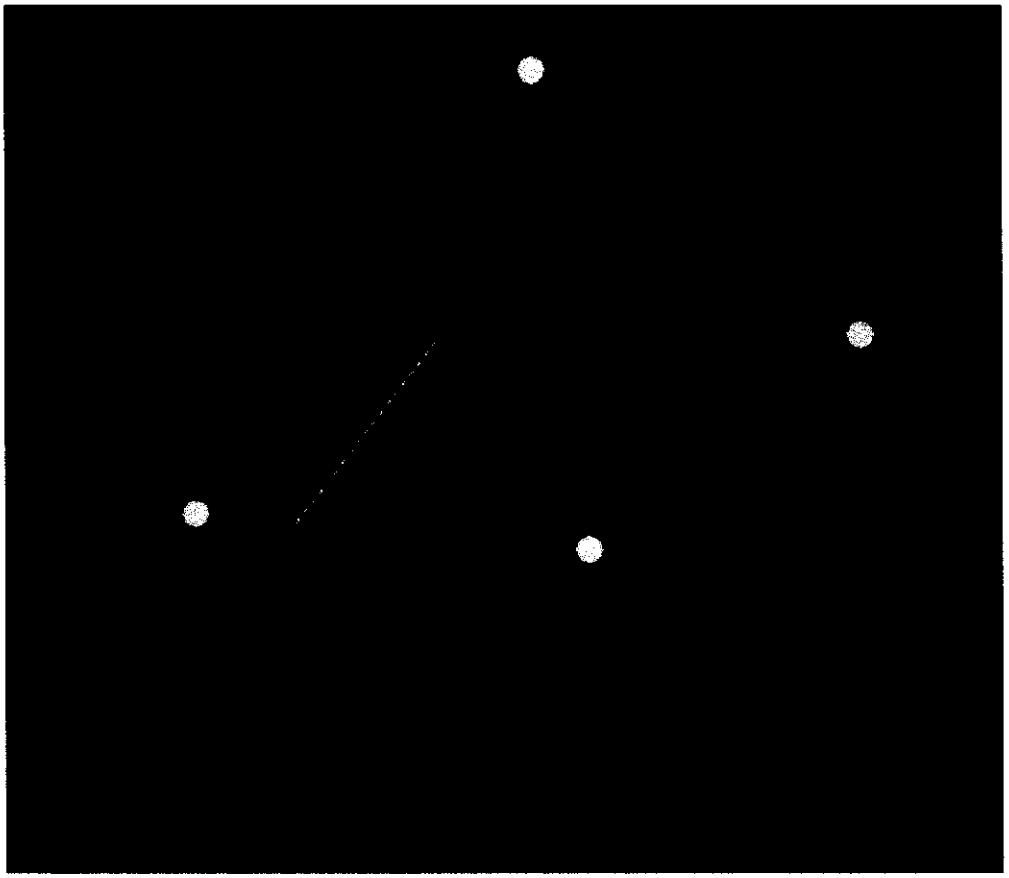
5.3.2.5 Classe EZobristDict

Dades generals			
Nom:	EZobristDict		
Tipus:	Entitat		
Descripció:	Entitat encarregada d'emmagatzemar tots els número aleatoris relacionats amb la generació de les claus de Zobrist per a les posicions segons l'algorisme descrit a l'apartat 2.3.1.2. Per motius d'eficiència, els atributs d'aquesta classe s'han fet públic evitant així l'overhead ocasionat per les crides a mètodes.		
Operacions més importants:	**Nom**	**Descripció**	**Diagrama**
	EZobristDict	Constructor de la classe que s'encarrega de la generació de tots els vectors de números aleatoris.	-
	initRand	Operació privada que inicialitza el generador de números aleatoris	-
	nextRand	Operació privada el pròxim número aleatori de la seqüència. Donat que initRand sempre utilitza la mateixa llavor la seqüència de números aleatoris sempre és la mateixa. Aquest fet és desitjable ja que si, per exemple, es volen guardar posicions a disc, les claus Zobrist coincidiran amb les claus de sessions posteriors	-
	updTauler	Actualitza les claus de Zobrist del tauler. Aquest mètode només es crida al principi(quan es configura la posició) ja que durant el procés de cerca les claus de Zobrist s'actualitzen dinàmicament.	-
	Nom	**Descripció**	
	KeysPeces	Emmagatzema un número aleatori per a cada casella, tipus de peça i color	
	KeysPecesLock	Idem que l'anterior per a detecció de col·lisions	
	KeysJugador	Un número aleatori per a cada jugador.	
	KeysJugadorLock	Idem que l'anterior per a detecció de col·lisions	
	KeysEnroc	Un número aleatori per jugador i flanc	
	KeysEnrocLock	Idem que l'anterior per a detecció de col·lisions	

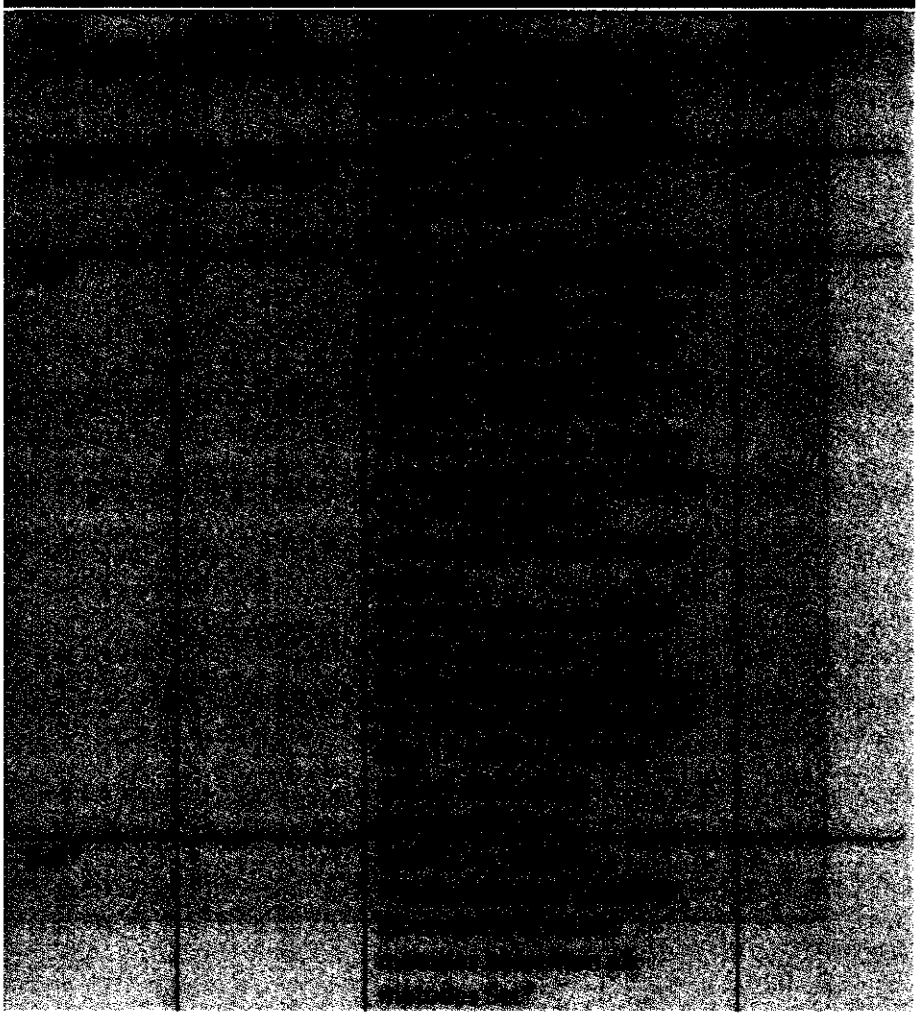
5.3.3 Disseny del component Avaluació de Posicions.

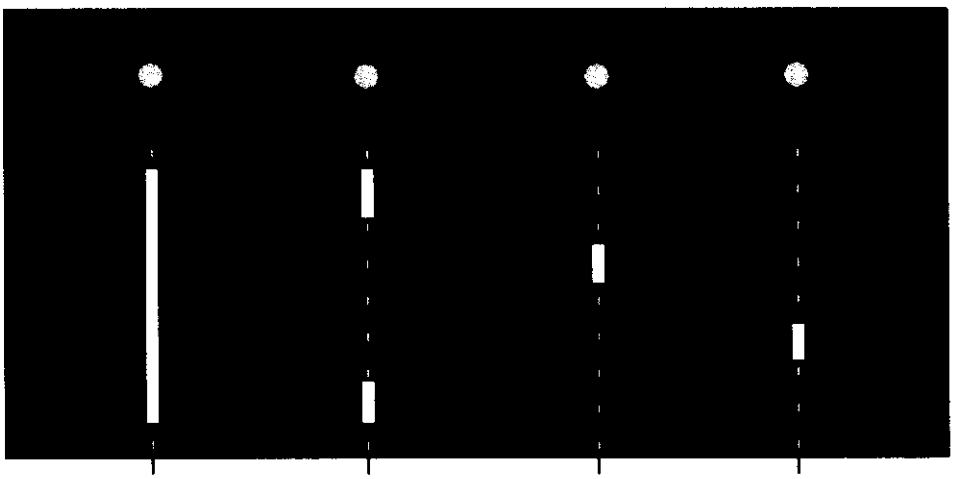
5.3.3.1 Estructura interna

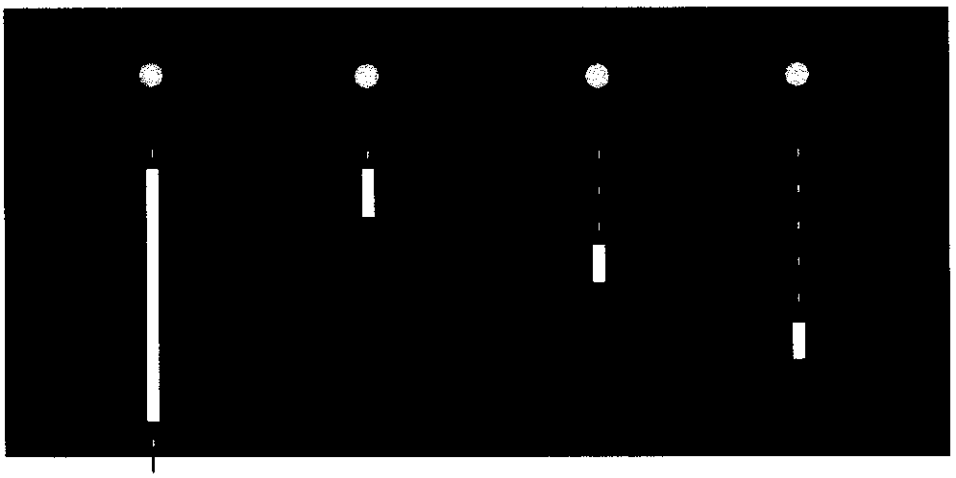
En el següent diagrama es presenta de manera més detallada l'estructura interna del component *Avaluació de Posicions* la funció principal del qual és la de fer l'avaluació de posicions mitjançant una funció heurística com la descrita a l'apartat 2.4.



5.3.3.2 Classe CmpAvalPosicions

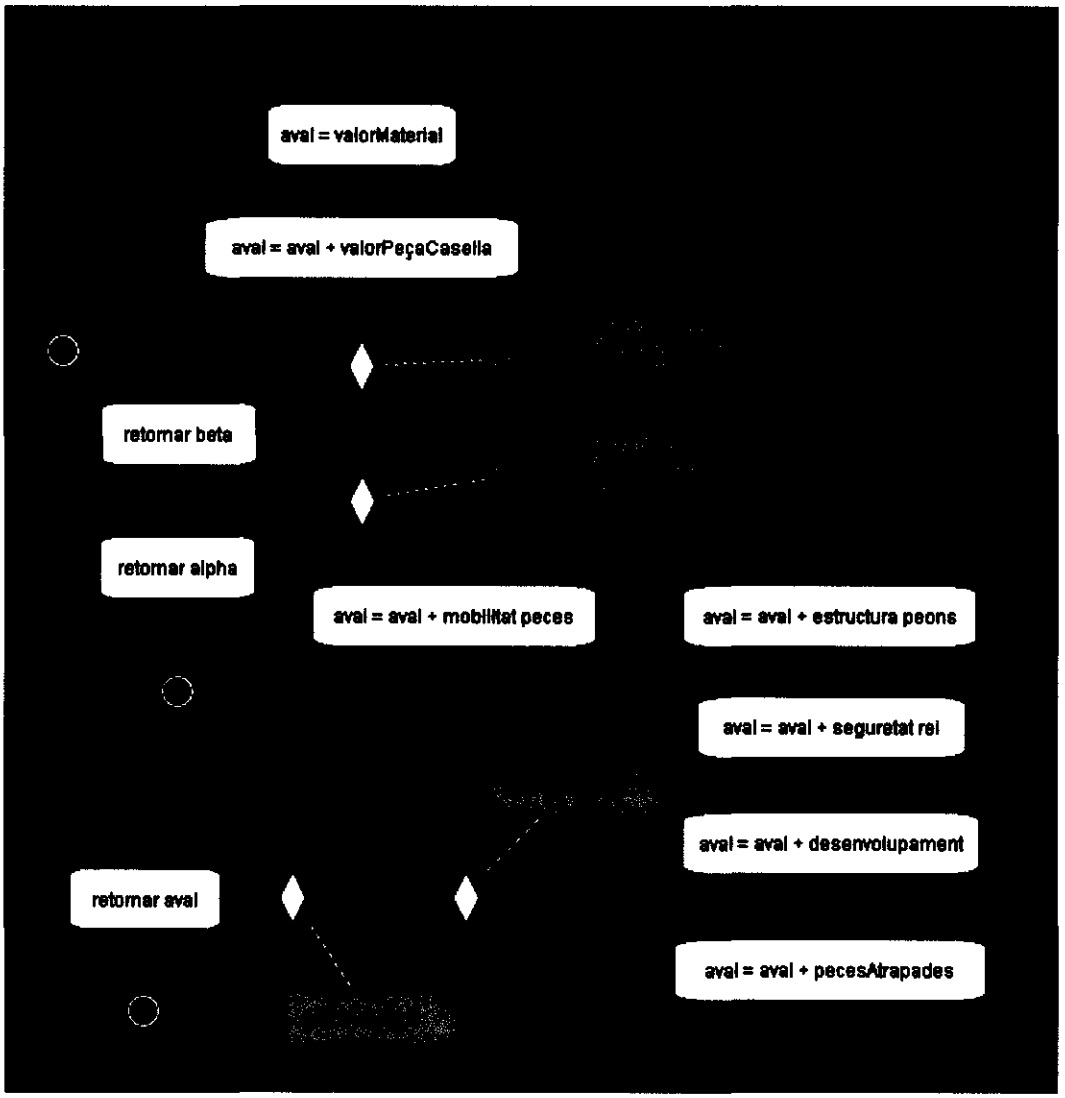
Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	





5.3.3.3 Classe BndEvalSenzill

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

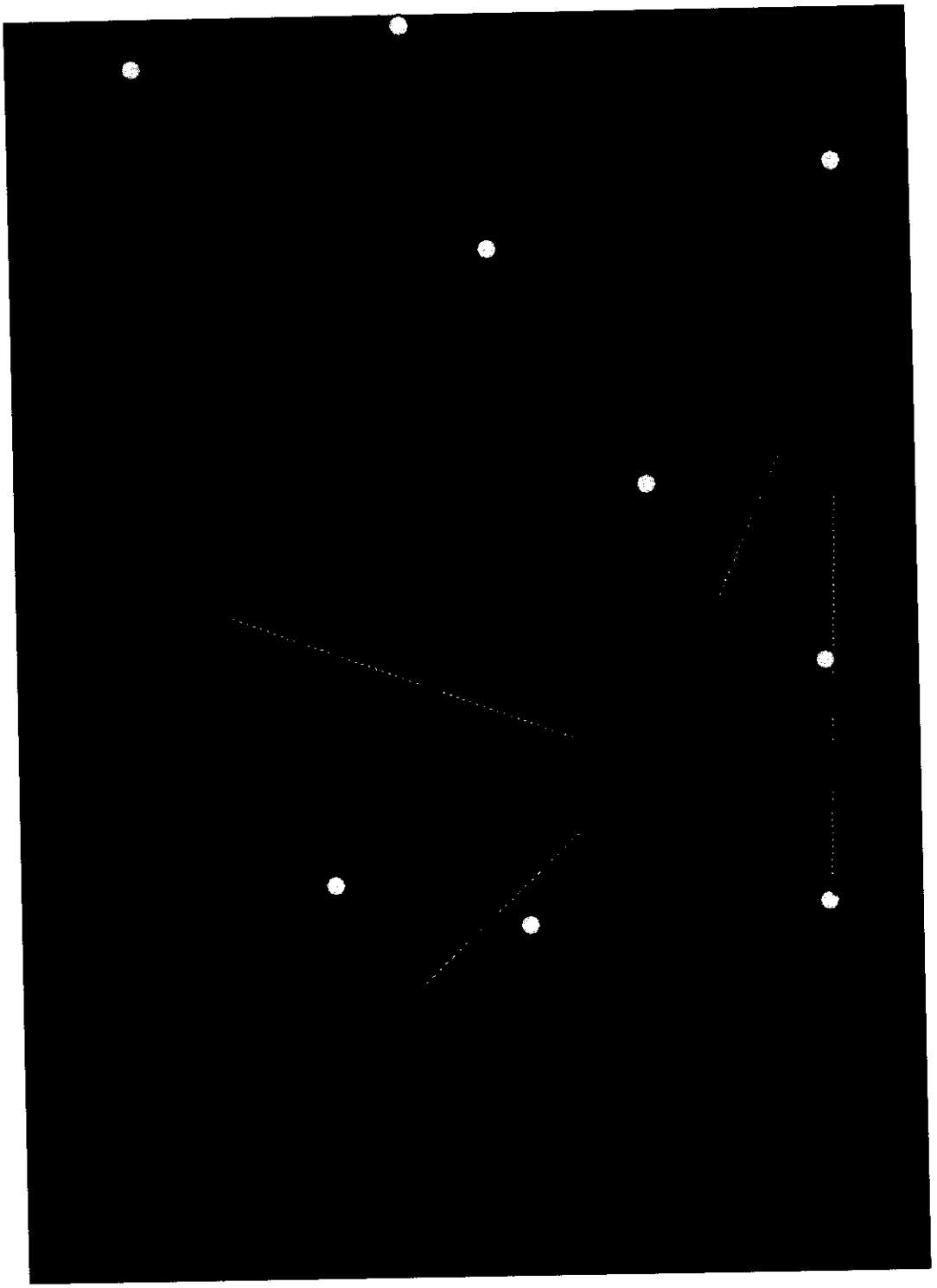


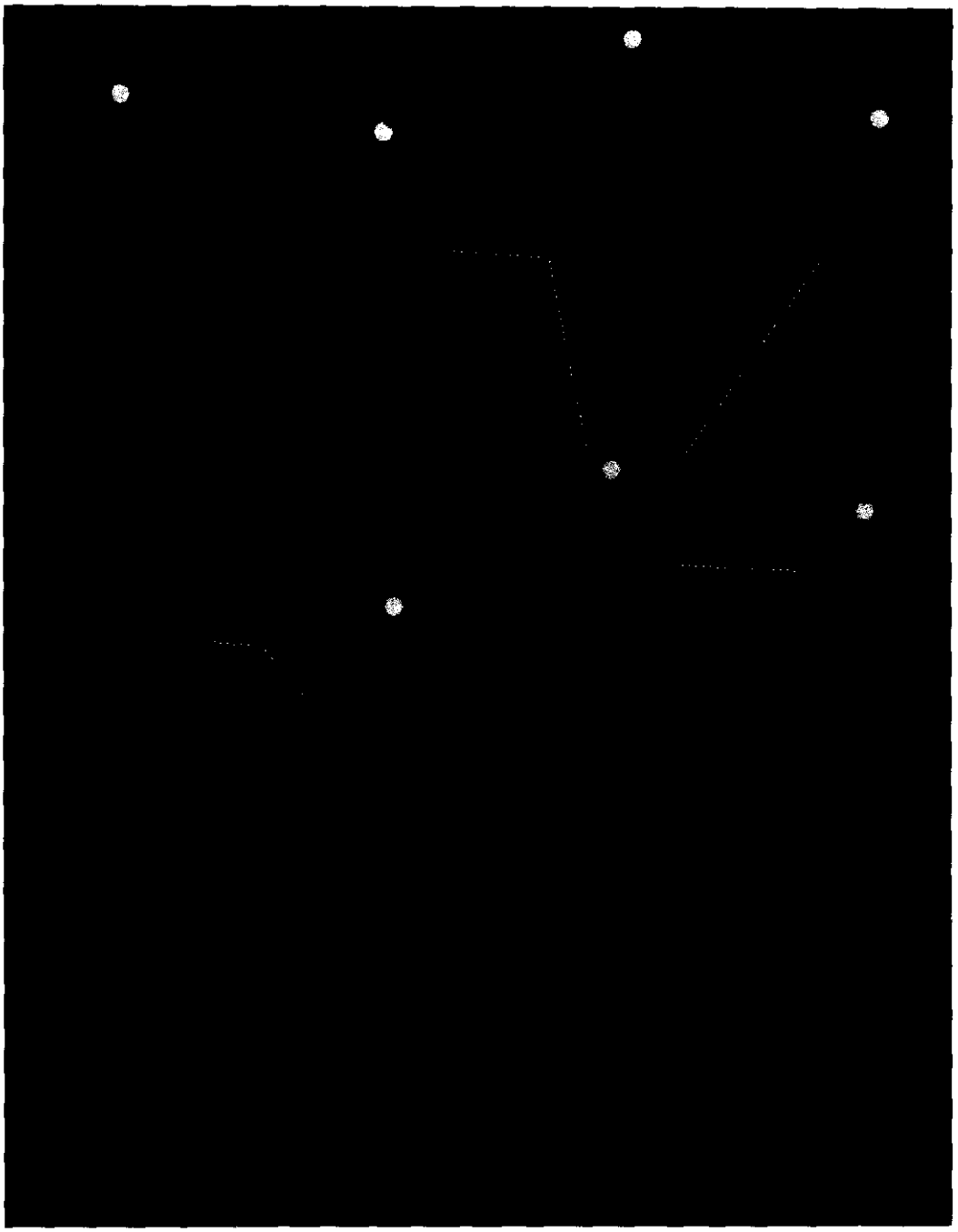
5.3.4 Disseny del component Generació Moviments

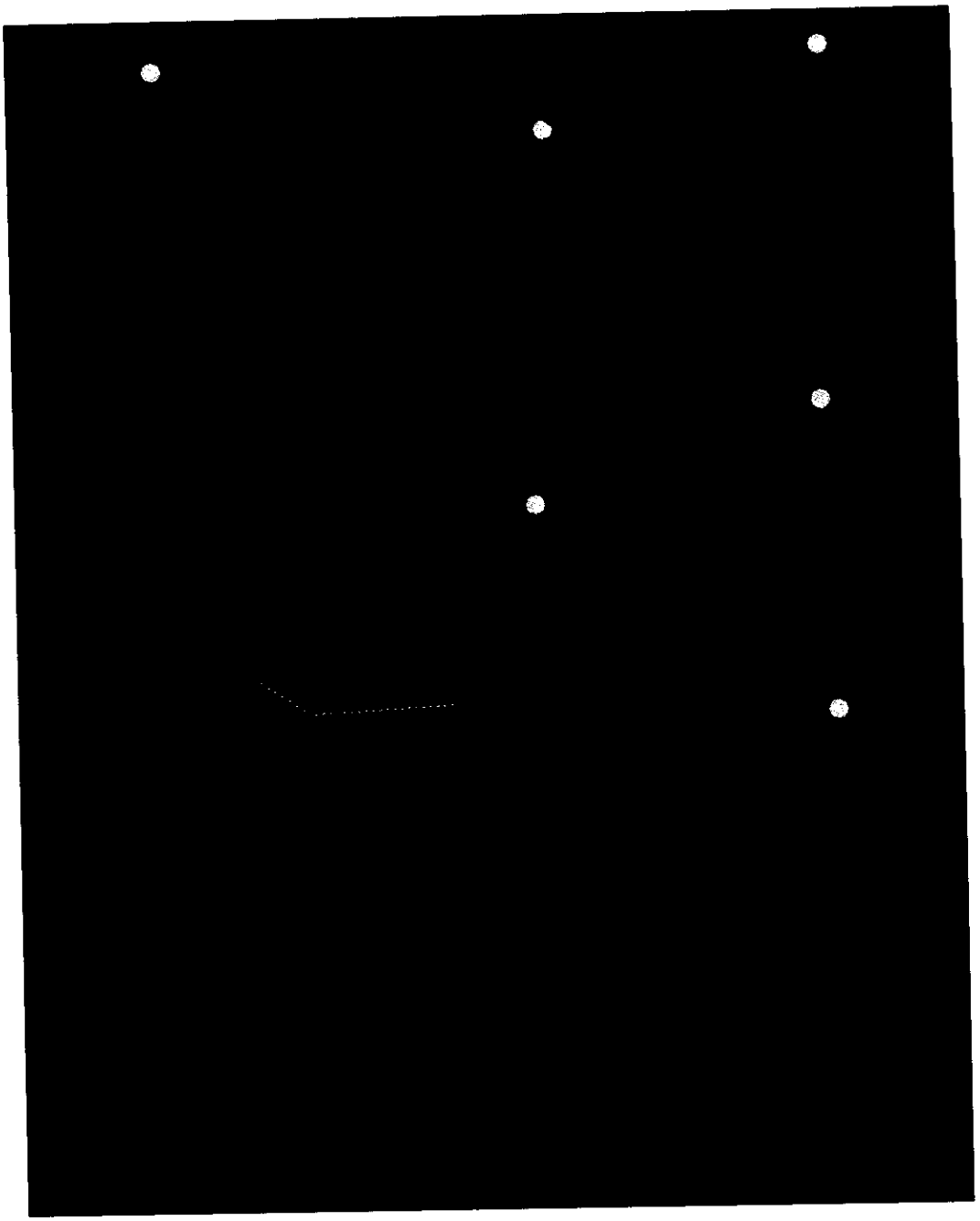
5.3.4.1 Estructura interna

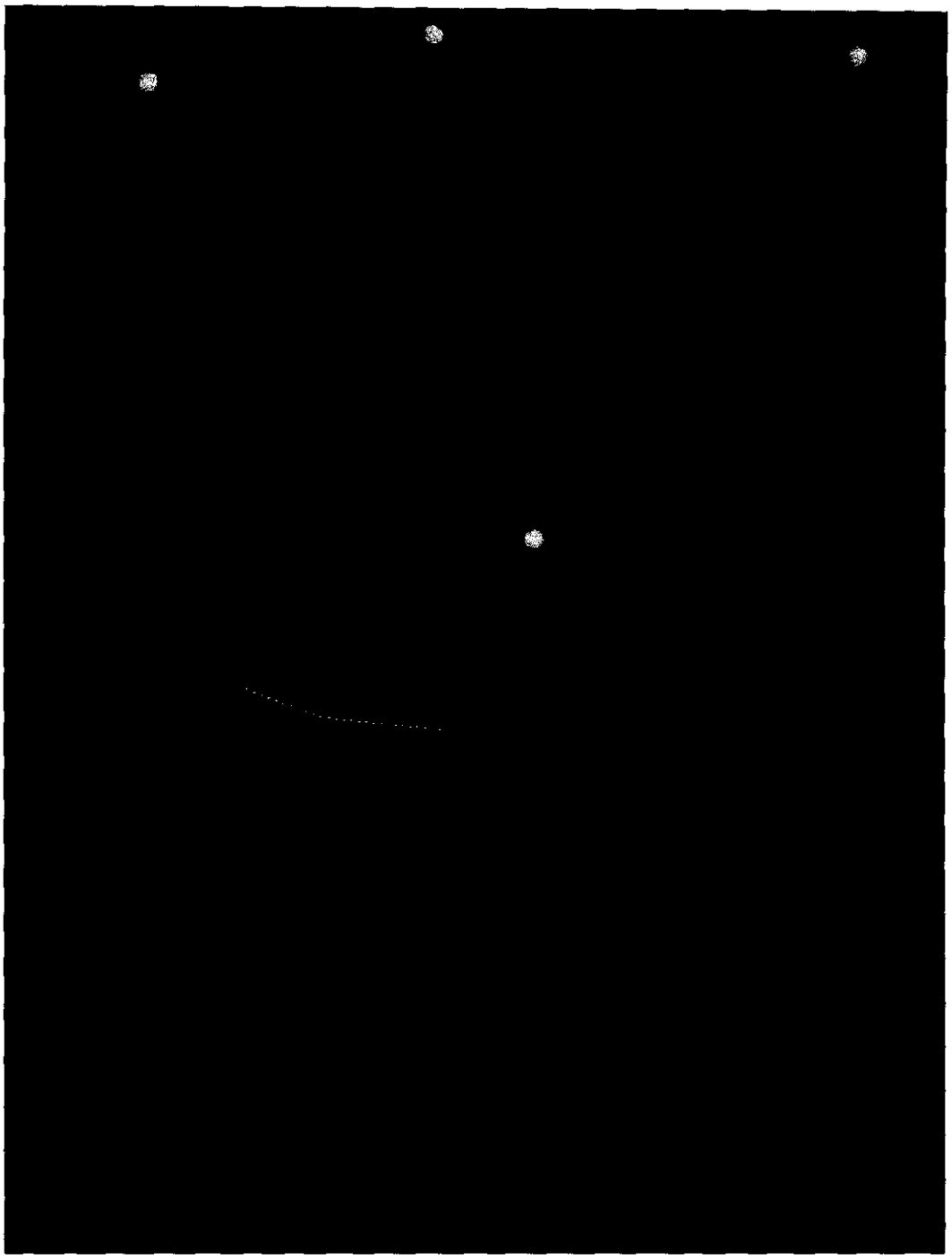
En el següents diagrames es presenta de manera detallada l'estructura interna del component *Generació de Moviments* la funció principal és la de generar els moviments legals que es poden jugar en una posició donada. En el primer diagrama s'il·lustra el mòdul en general i en la resta

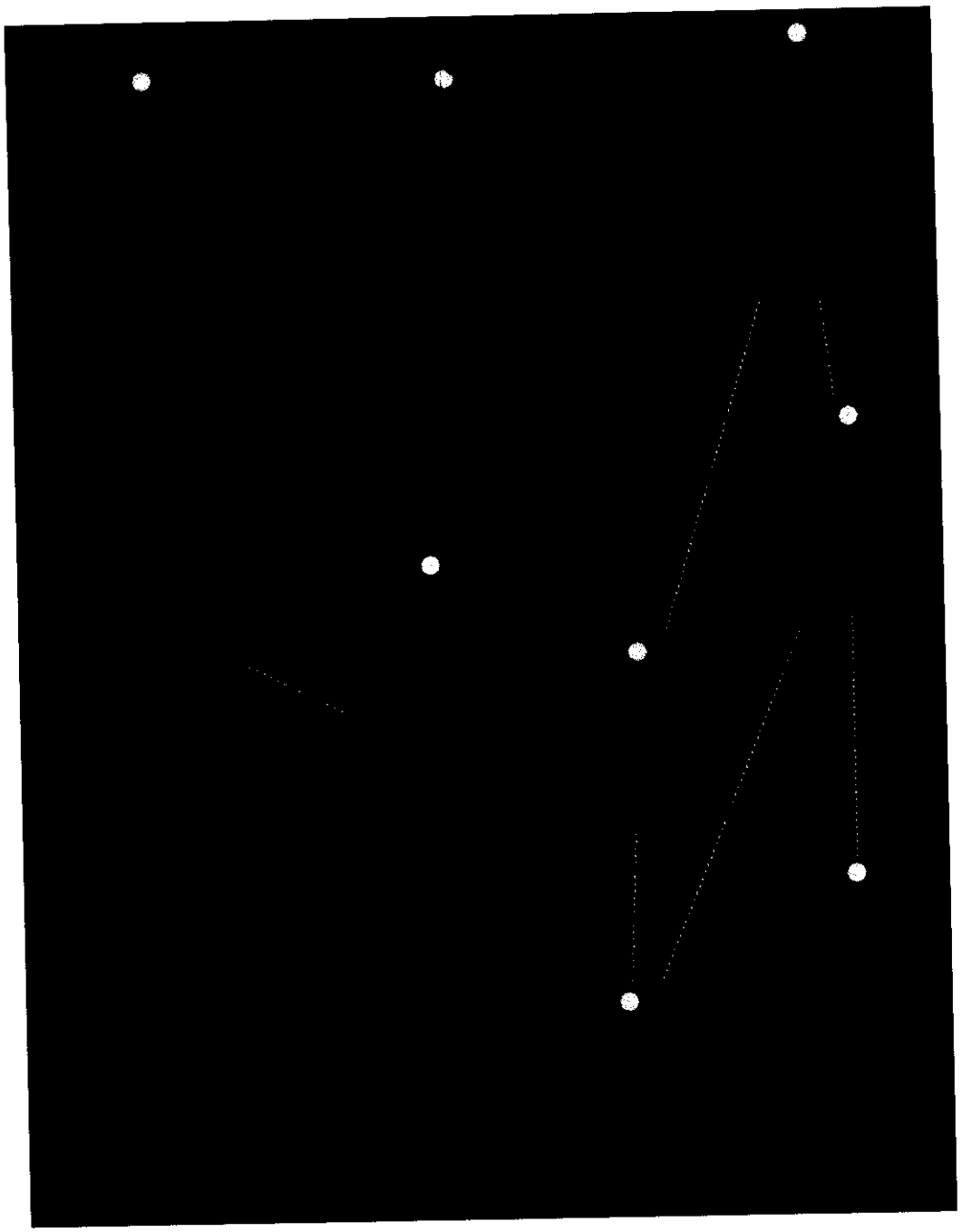
de diagrames es detallen les classes específiques involucrades en la generació dels moviments de cada peça.

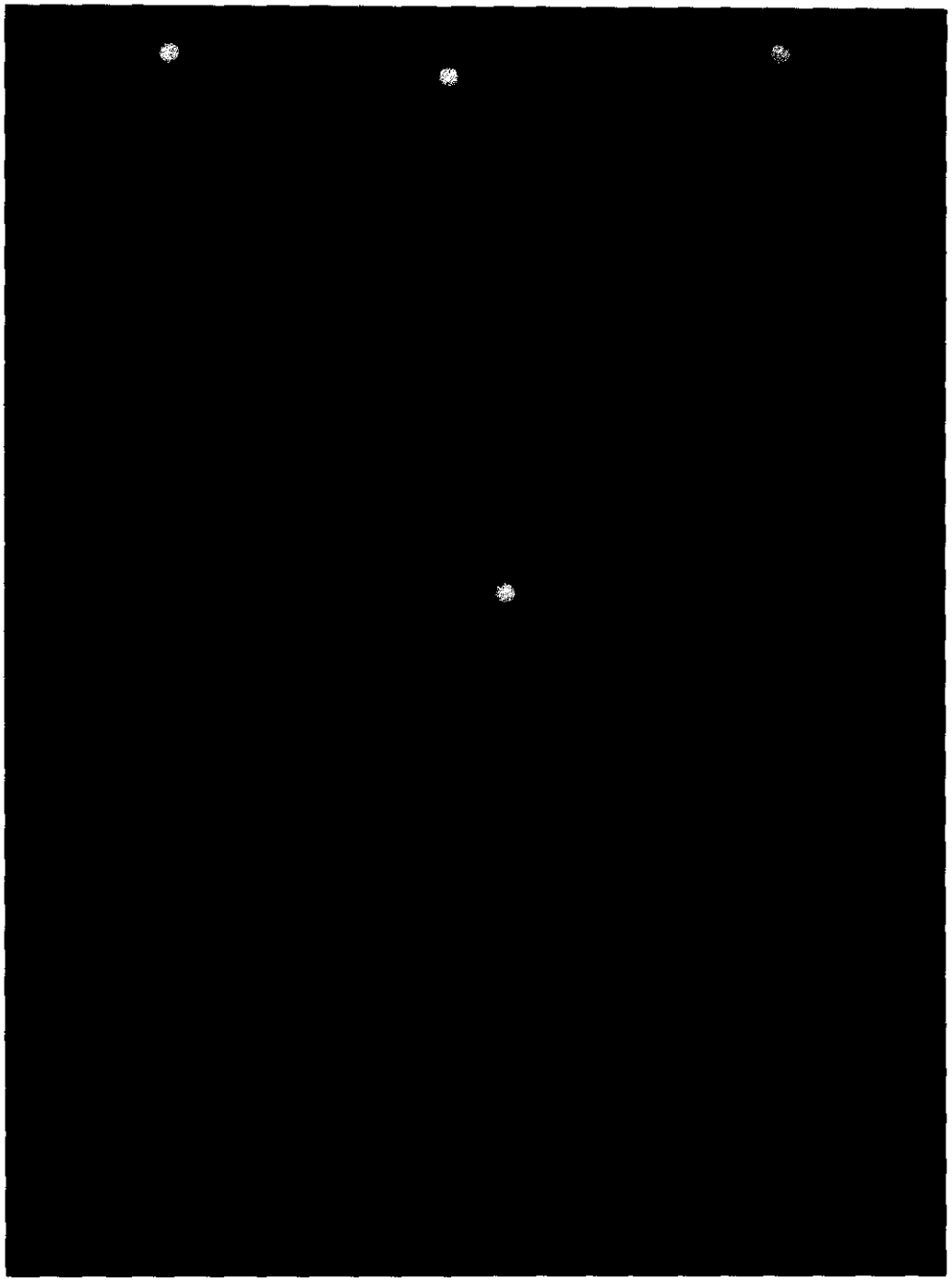


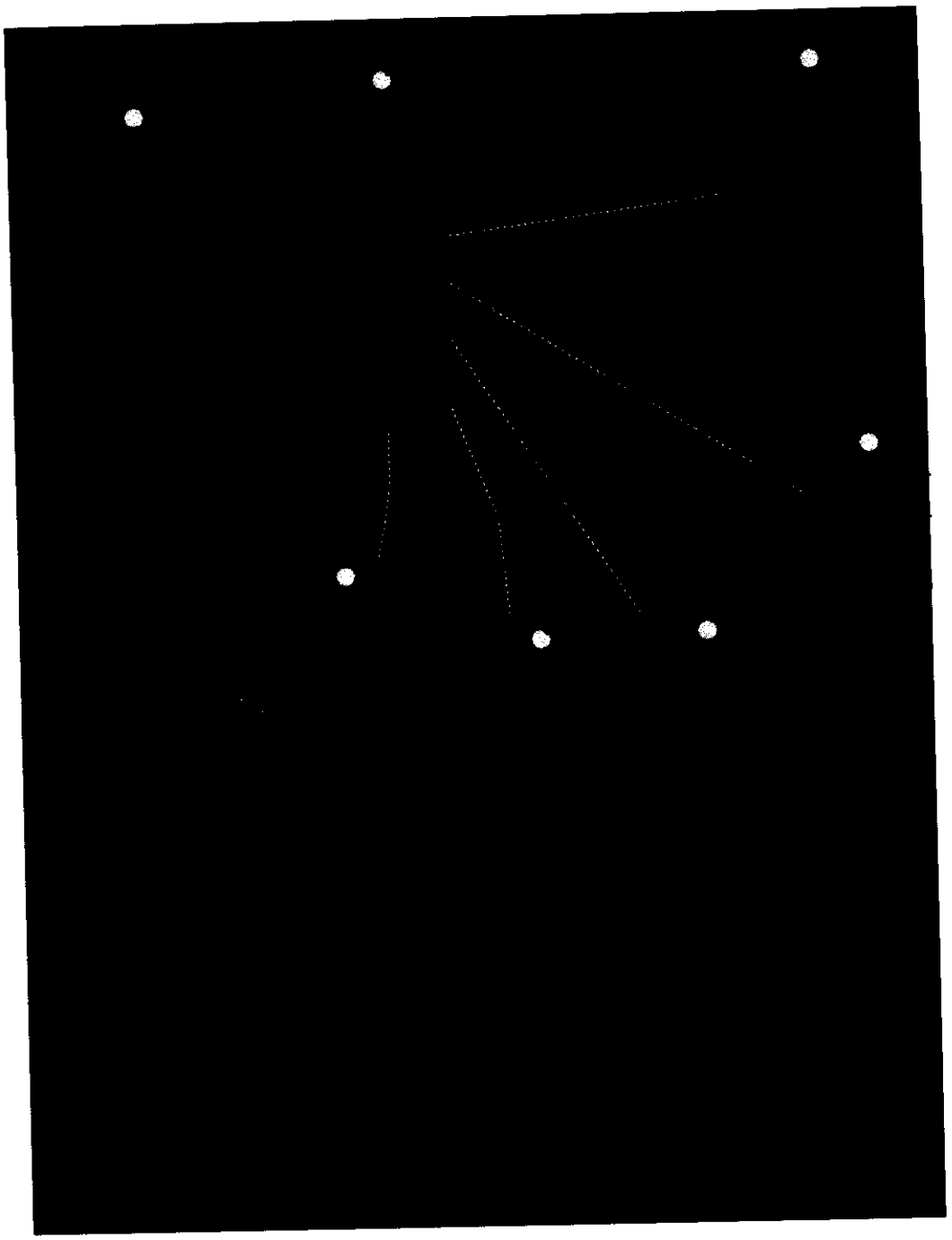




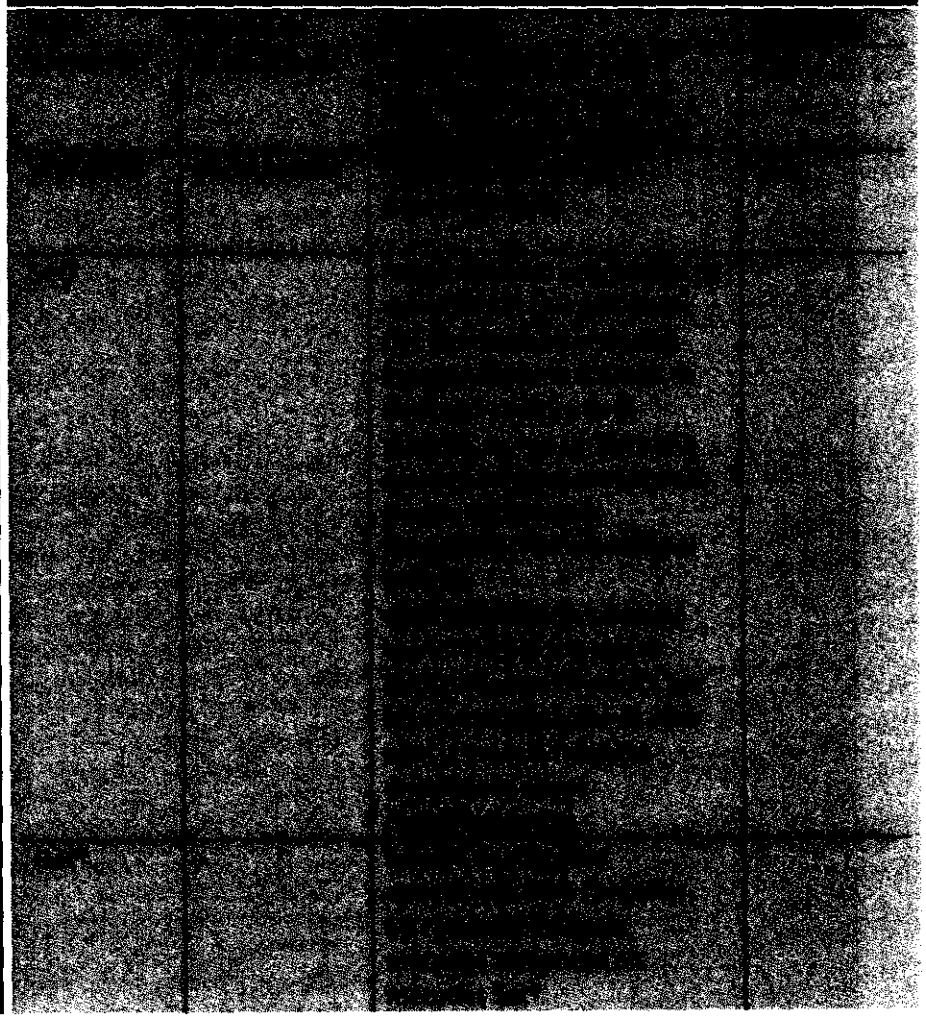


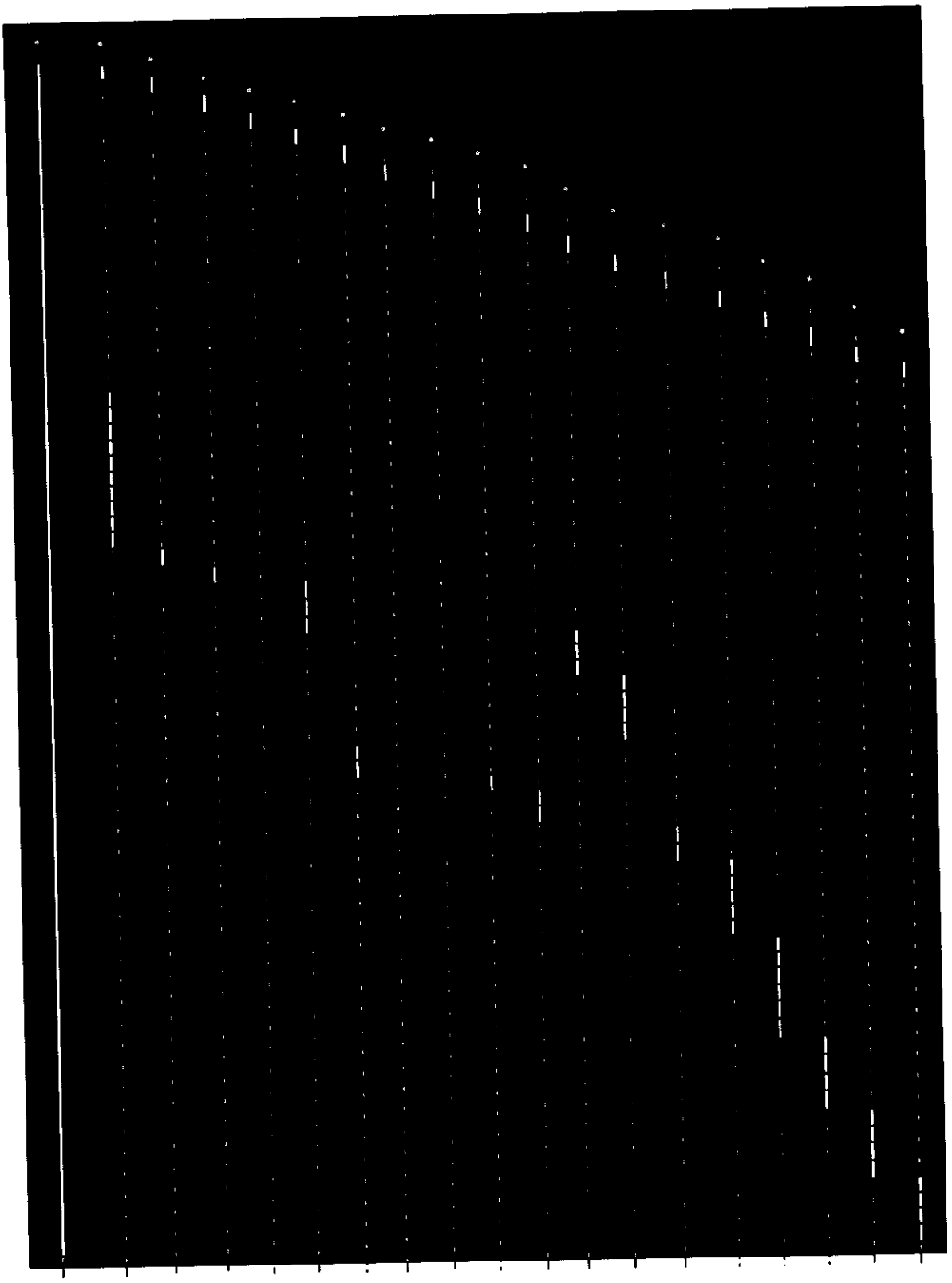


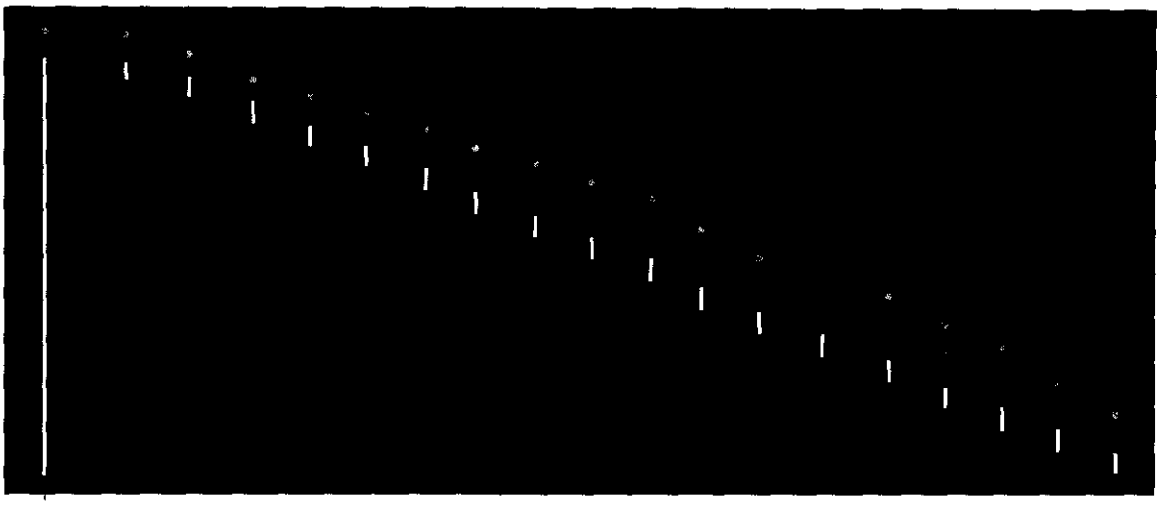




5.3.4.2 Classe CmpGenMovs

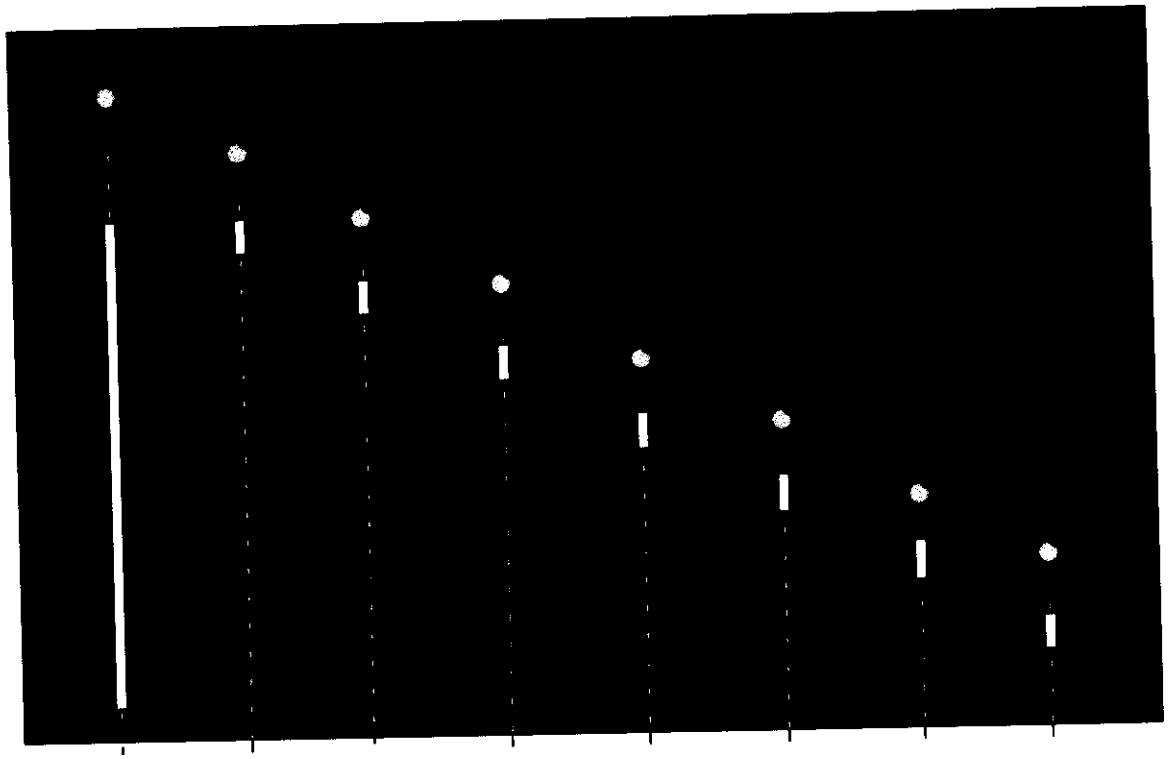
Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

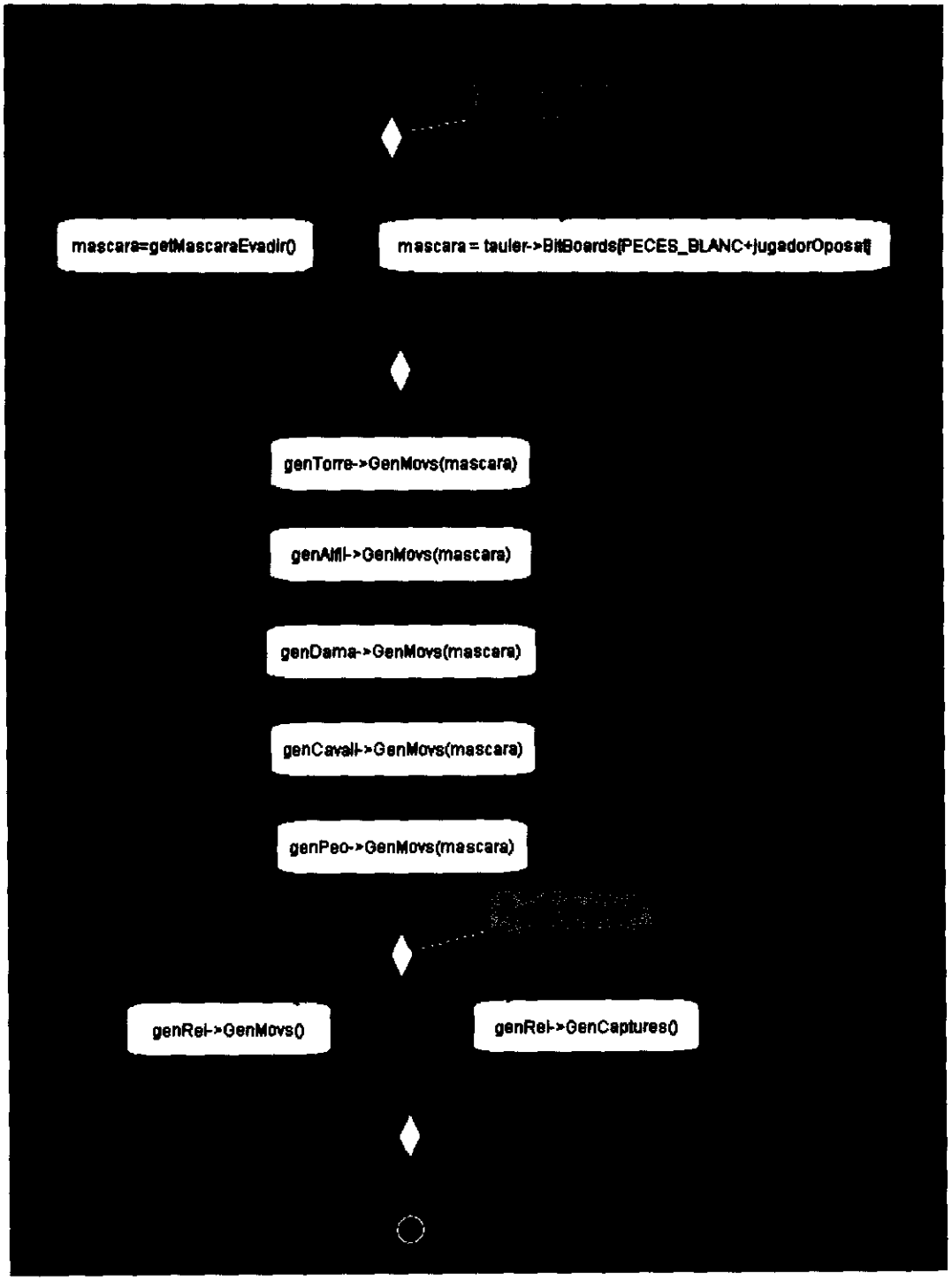


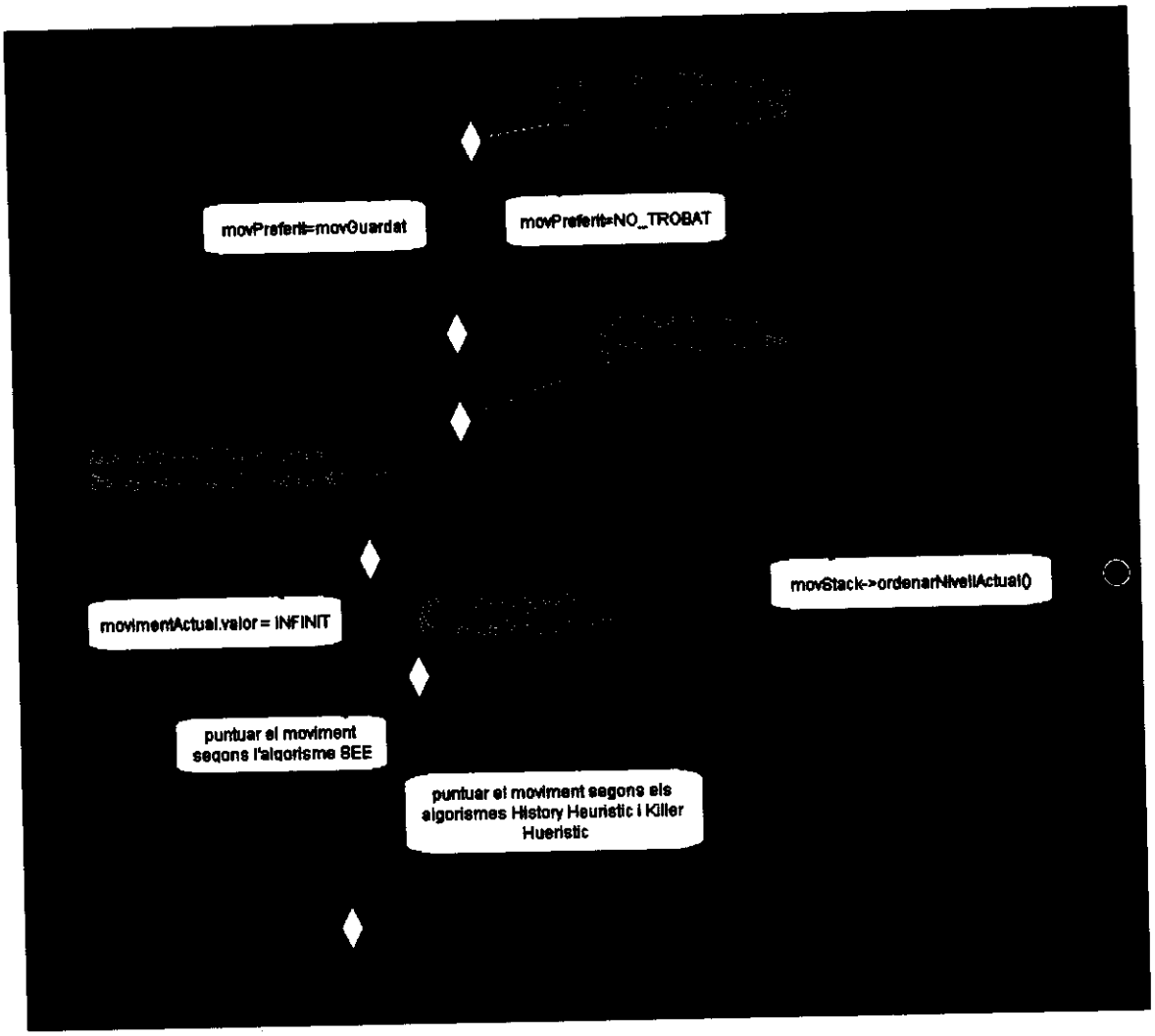


5.3.4.3 Classe BndGeneracioMoviments

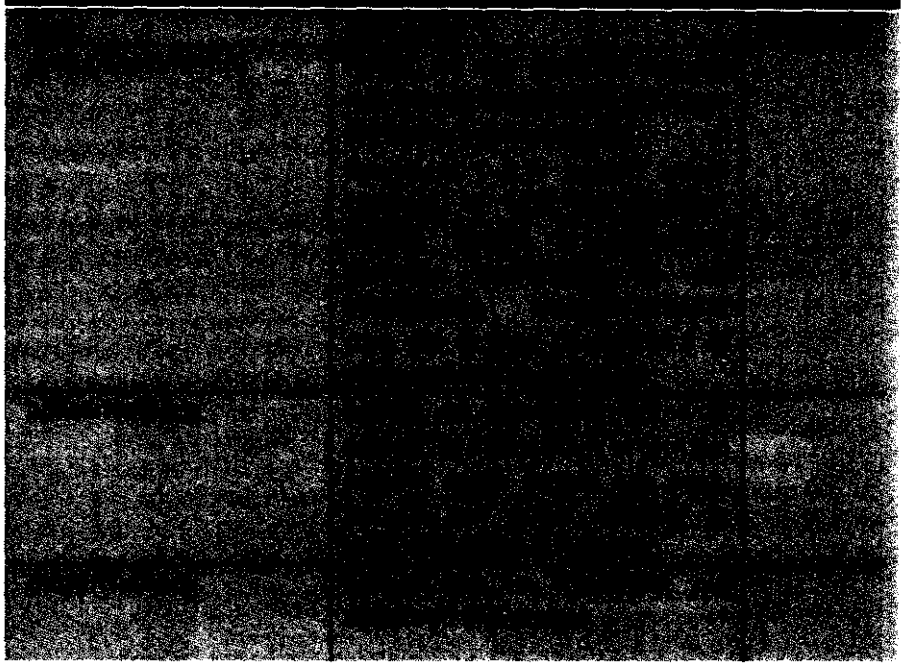
Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	



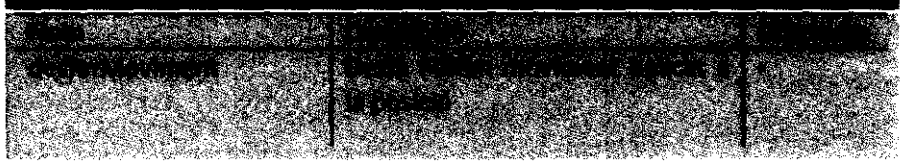

mascara=getMascaraEvadir()
mascara = tauler->BitBoards[PECES_BLANC+jugadorOposat]
genTorre->GenMovs(mascara)
genAlfil->GenMovs(mascara)
genDama->GenMovs(mascara)
genCavall->GenMovs(mascara)
genPeo->GenMovs(mascara)
genRei->GenMovs()
genRei->GenCaptures()


movPreferit=movGuardat
movPreferit=NO_TROBAT
movStack->ordenarNivellActual()
movimentActual.valor = INFINIT
puntuar el moviment segons l'algorisme BEE
puntuar el moviment segons els algorismes History Heuristic i Killer Hueristic

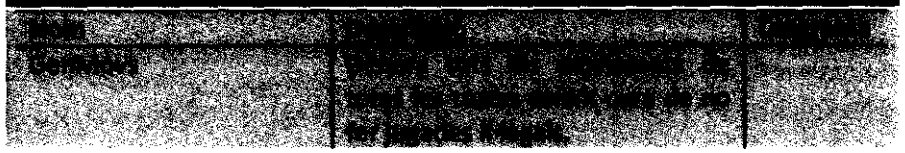
5.3.4.4 Classe BndAplicarMoviment

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.4.5 Classe BndDesferMoviment

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.4.6 Classe CntGenMovsTorre

Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

5.3.4.7 Classe CntGenMovsAlfil

Dades generals			
Nom:	CntGenMovsAlfil		
Tipus:	Control		
Descripció:	Classe especialitzada en la generació de tots el moviments de tots els alfils		
Operacions més importants:	**Nom**	**Descripció**	**Diagrama**
	GenMovs	Genera tots els moviments de tots els alfils tenint cura de no fer jugades il·legals.	-

5.3.4.8 Classe CntGenMovsDama

Dades generals			
Nom:	CntGenMovsDama		
Tipus:	Control		
Descripció:	Classe especialitzada en la generació de tots el moviments de totes les dames		
Operacions més importants:	**Nom**	**Descripció**	**Diagrama**
	GenMovs	Genera tots els moviments de totes les dames tenint cura de no fer jugades il·legals.	-

5.3.4.9 Classe CntGenMovsCavall

Dades generals			
Nom:	CntGenMovsCavall		
Tipus:	Control		
Descripció:	Classe especialitzada en la generació de tots el moviments de tots els cavalls		
Operacions més importants:	**Nom**	**Descripció**	**Diagrama**
	GenMovs	Genera tots els moviments de tots els cavalls tenint cura de no fer jugades il·legals.	-

5.3.4.10 Classe CntGenMovsPeo

Dades generals			
Nom:	CntGenMovsPeo		
Tipus:	Control		
Descripció:	Classe especialitzada en la generació de tots el moviments de tots els peons		
Operacions més importants:	**Nom**	**Descripció**	**Diagrama**
	GenMovs	Genera tots els moviments de tots els peons tenint cura de no fer jugades il·legals.	-

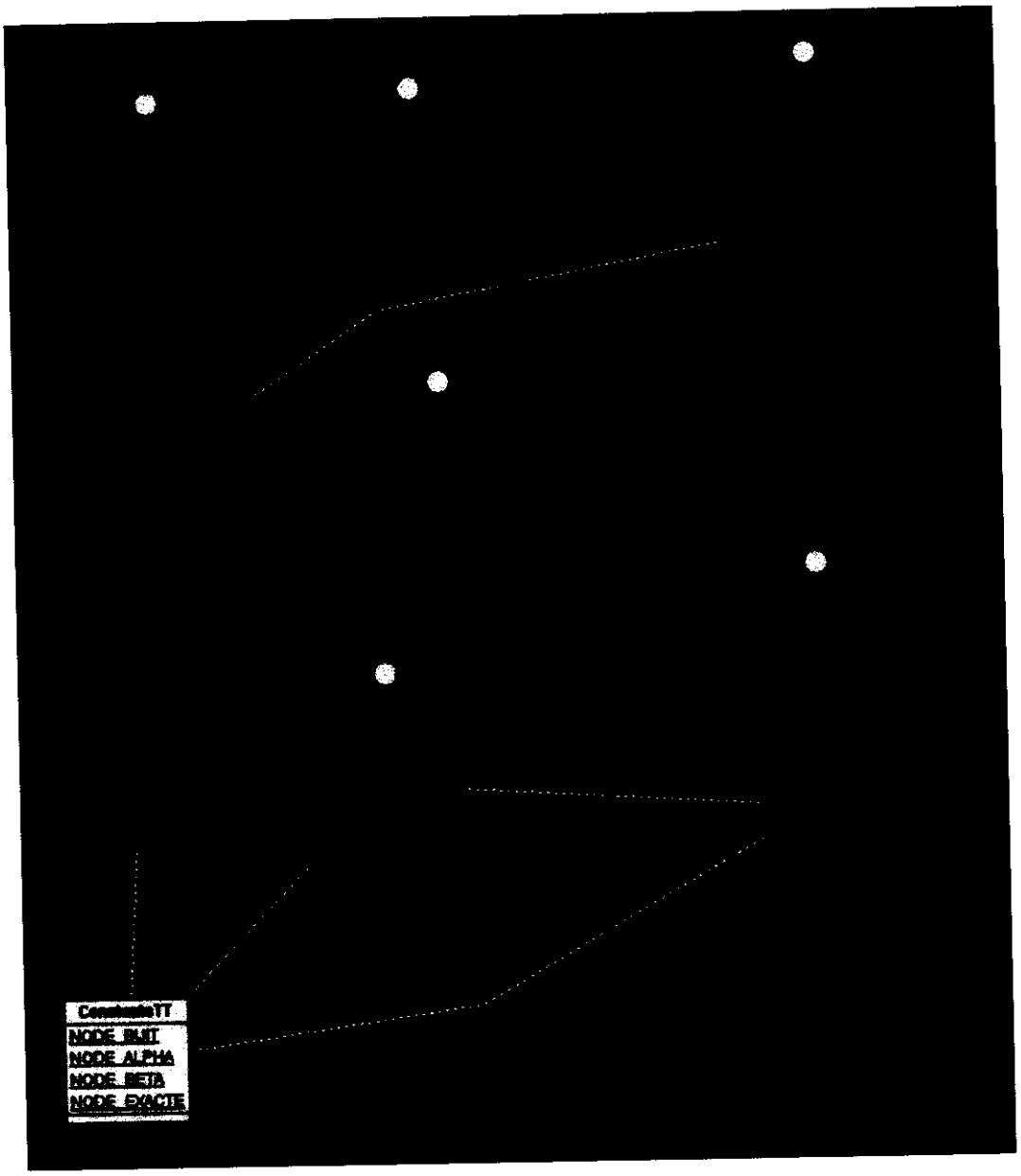
5.3.4.11 Classe CntGenMovsRei

Dades generals			
Nom:	CntGenMovsPeo		
Tipus:	Control		
Descripció:	Classe especialitzada en la generació de tots el moviments del rei		
Operacions més importants:	**Nom**	**Descripció**	**Diagrama**
	GenMovs	Genera tots els moviments del rei tenint cura de no fer jugades il·legals.	-

5.3.5 Disseny del component Algorísmica de Cerca

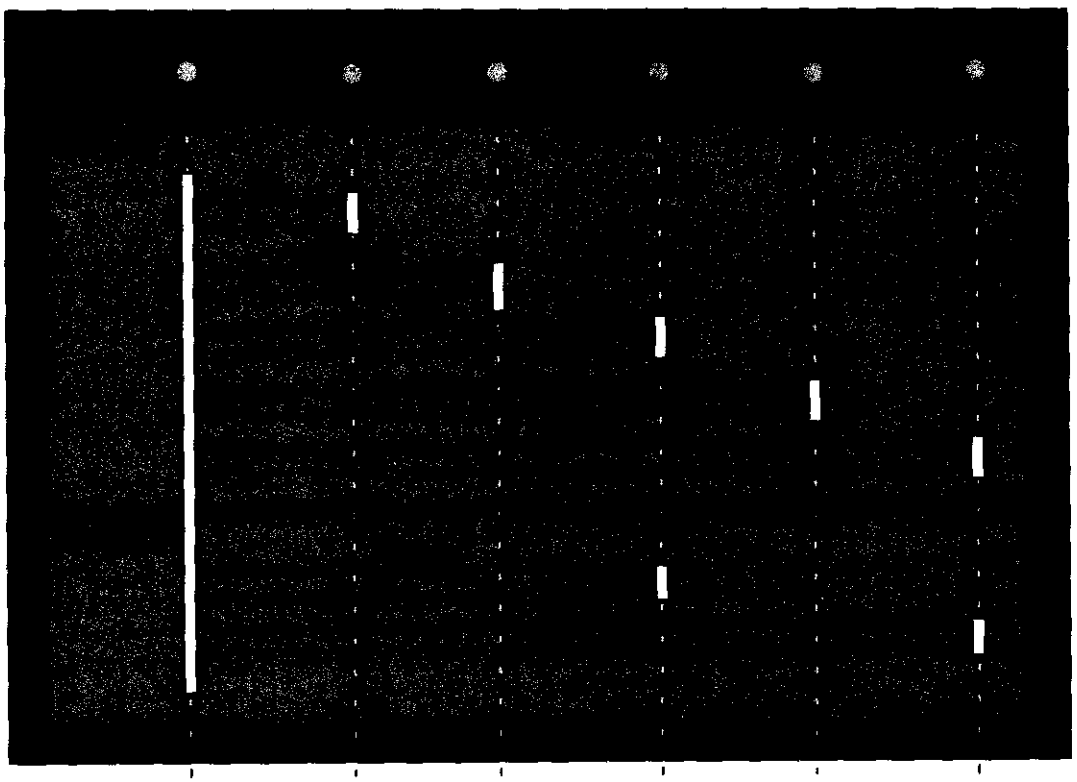
5.3.5.1 Estructura interna

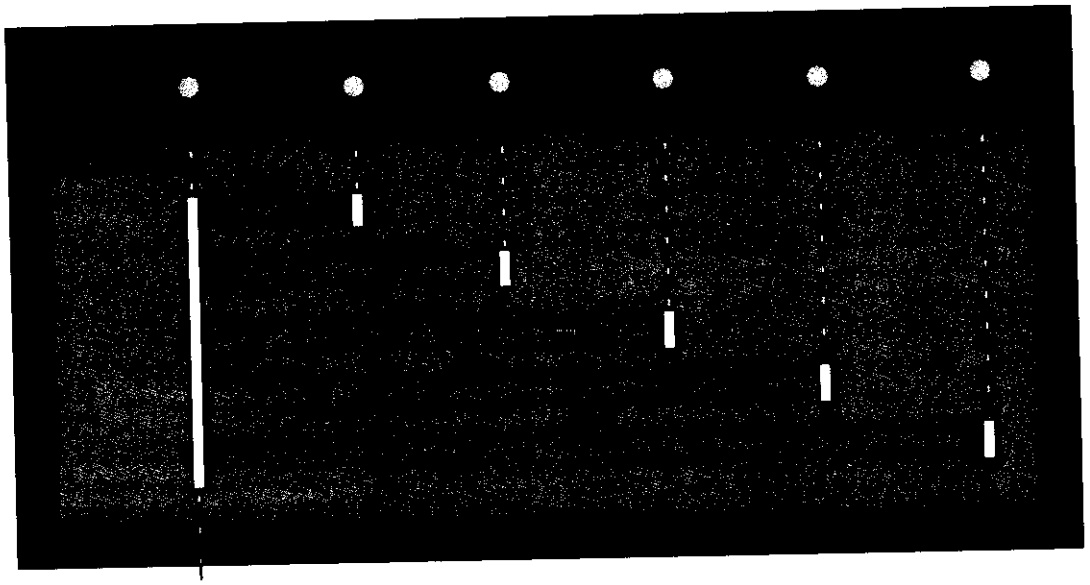
Aquest component conté, llevat de l'algorisme de profunditat iterativa, tots els objectes i mètodes referents a l'algorísmica de Cerca. En el següent diagrama es poden veure les classes que el composen i la seva estructura interna:



5.3.5.2 Classe CmpCerca

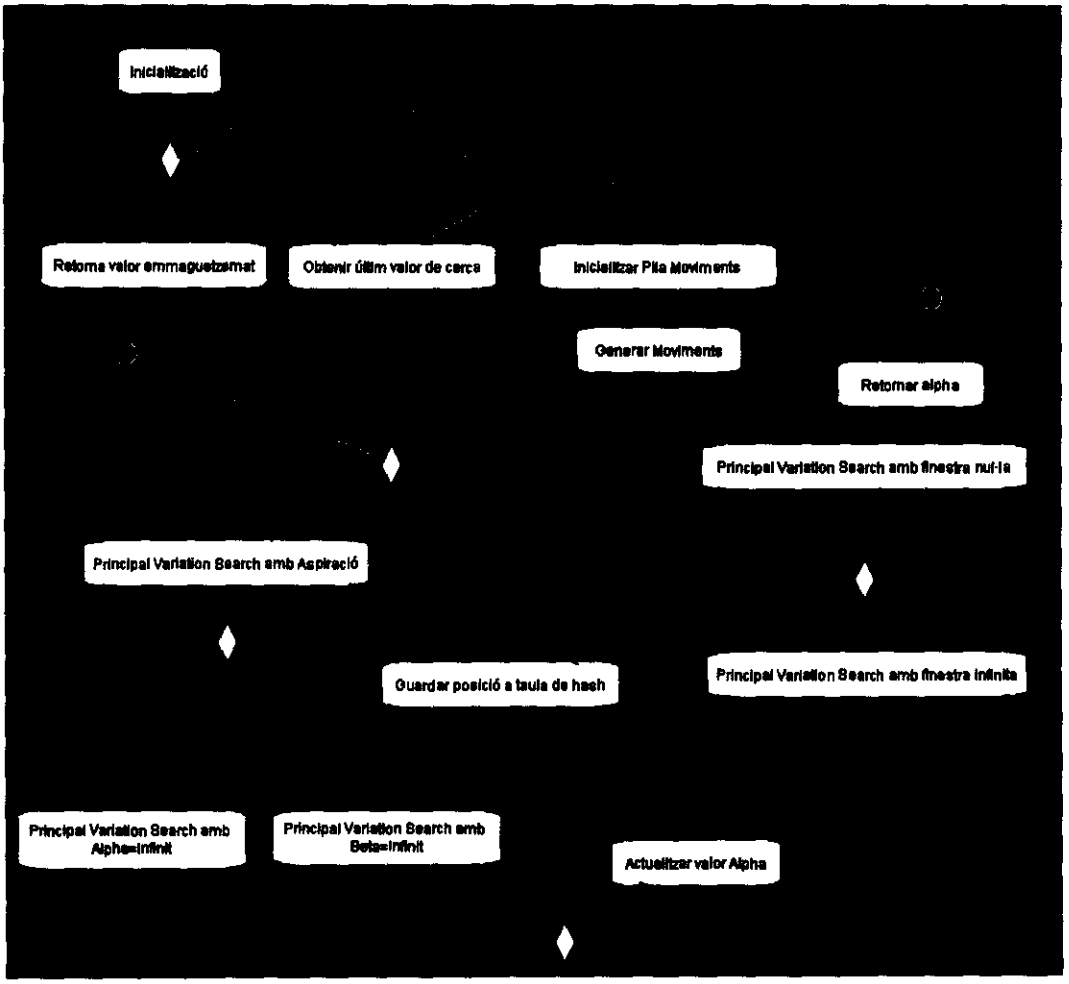
Dades generals	
Nom:	
Tipus:	
Descripció:	
Operacions més importants:	

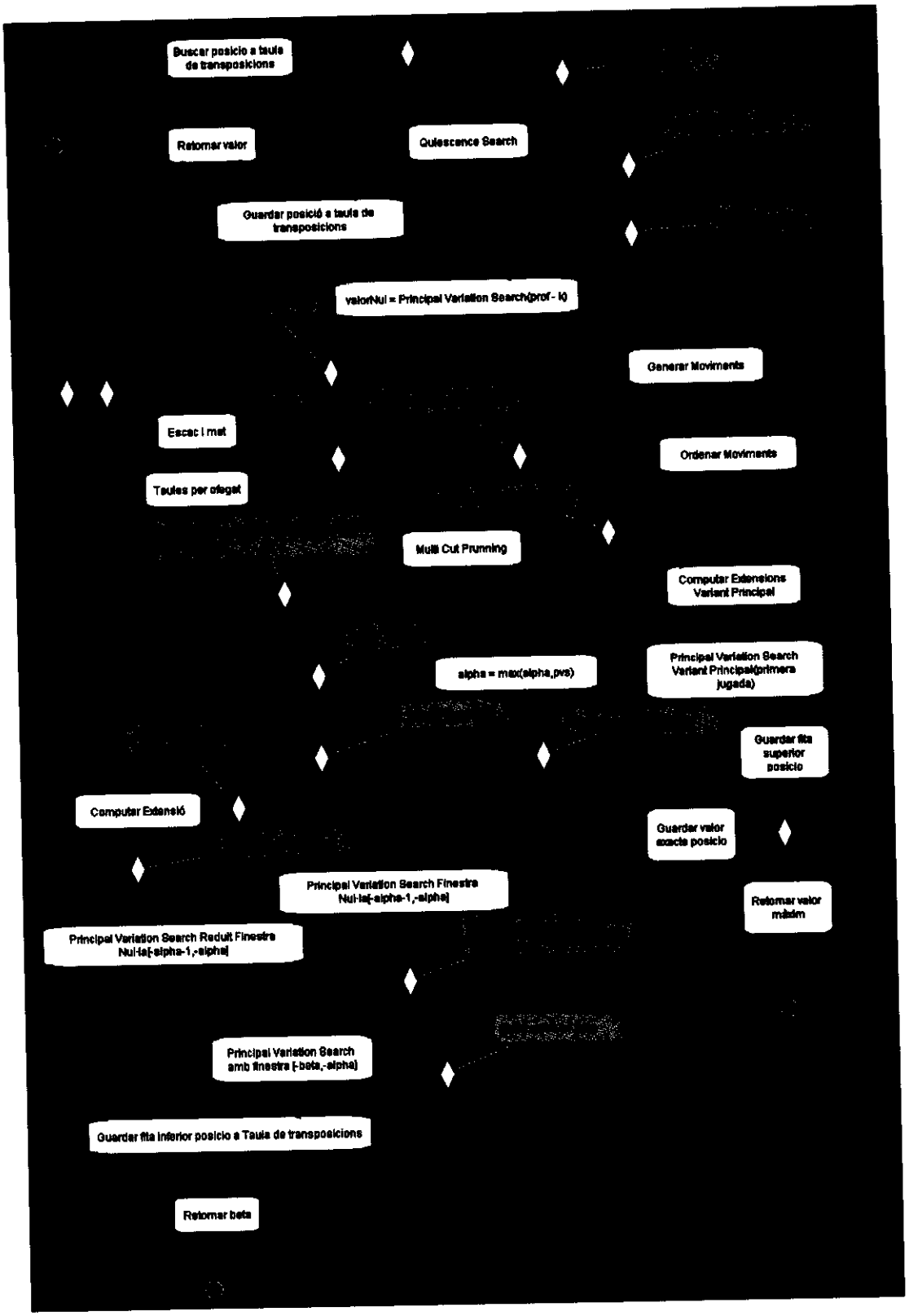


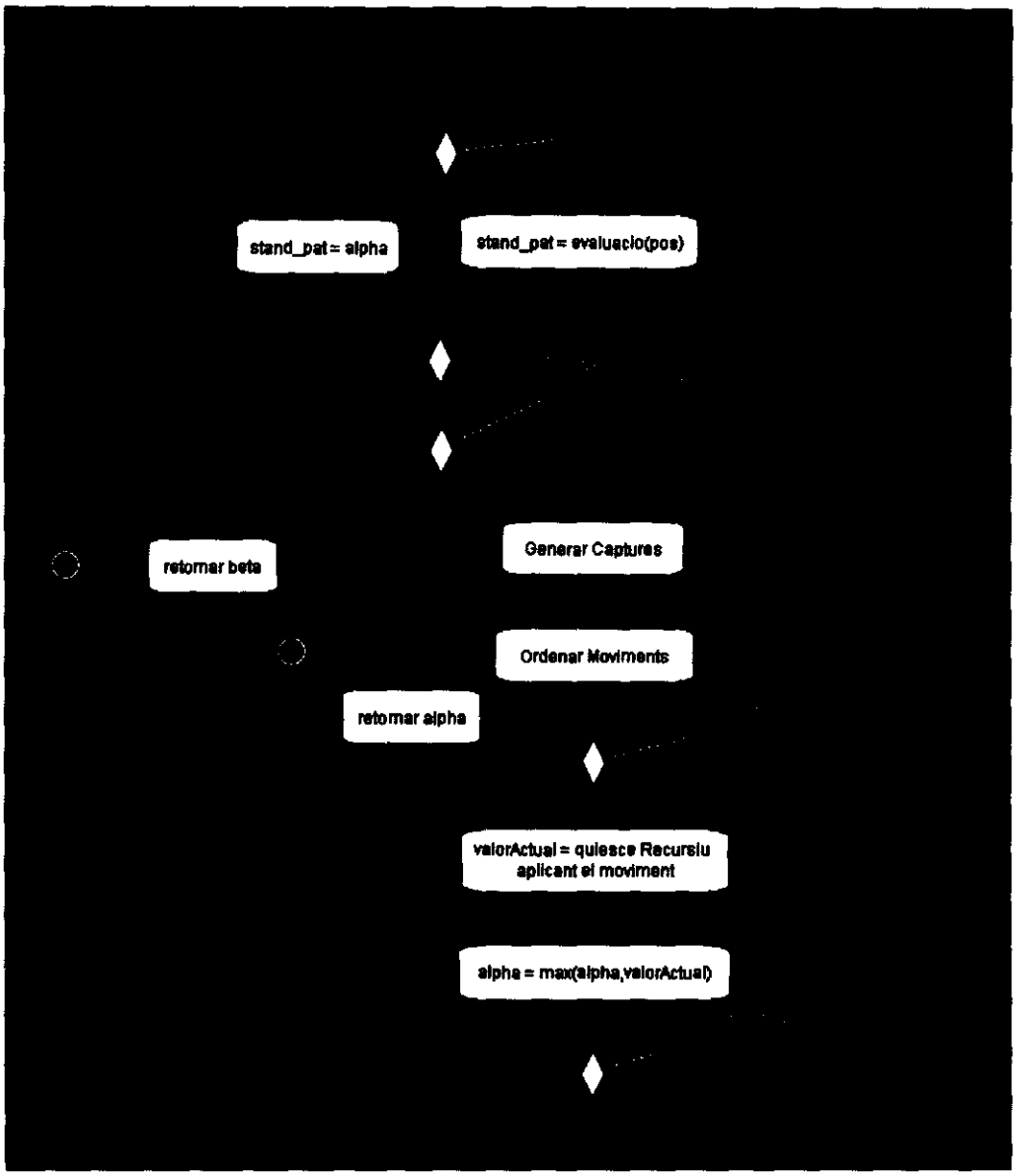


5.3.5.3 Classe BndAlphaBetaTT

Dades generals			
Nom:			
Tipus:			
Descripció:			
Operacions més importants:	[illegible]	[illegible]	[illegible]
	PVS	Retorna la nova de la posició arrel, és a dir, la primera acció del vector en la qual s'esdevenen els dos jugadors.	A4
	PV	Funció recursiva que implementa l'algorisme Principal Variation Searching(PVS) les mateixes descrites a l'apartat 2.4) a la qual cada PVS des de cadascuna de les posicions filles de la posició arrel.	A5
	Quiescence	Funció recursiva que implementa l'algorisme Quiescence Search vist a l'apartat 2.3.	A6


Inicialització
Retorna valor emmagatzemat
Obtenir últim valor de cerca
Inicialitzar Pila Moviments
Generar Moviments
Retornar alpha
Principal Variation Search amb finestra nul·la
Principal Variation Search amb Aspiració
Guardar posició a taula de hash
Principal Variation Search amb finestra infinita
Principal Variation Search amb Alpha=infinit
Principal Variation Search amb Beta=infinit
Actualitzar valor Alpha


Buscar posicio a taula de transposicions
Retornar valor
Quiescence Search
Guardar posició a taula de transposicions
valorNul = Principal Variation Search(prof - k)
Generar Moviments
Escac i mat
Ordenar Moviments
Taules per ofegat
Multi Cut Prunning
Computar Extensions Variant Principal
alpha = max(alpha,pvs)
Principal Variation Search Variant Principal(primera jugada)
Guardar fita superior posicio
Computar Extensió
Guardar valor exacte posicio
Principal Variation Search Finestra Nul·la[-alpha-1,-alpha]
Retornar valor maxim
Principal Variation Search Reduit Finestra Nul·la[-alpha-1,-alpha]
Principal Variation Search amb finestra [-beta,-alpha]
Guardar fita inferior posicio a Taula de transposicions
Retornar beta

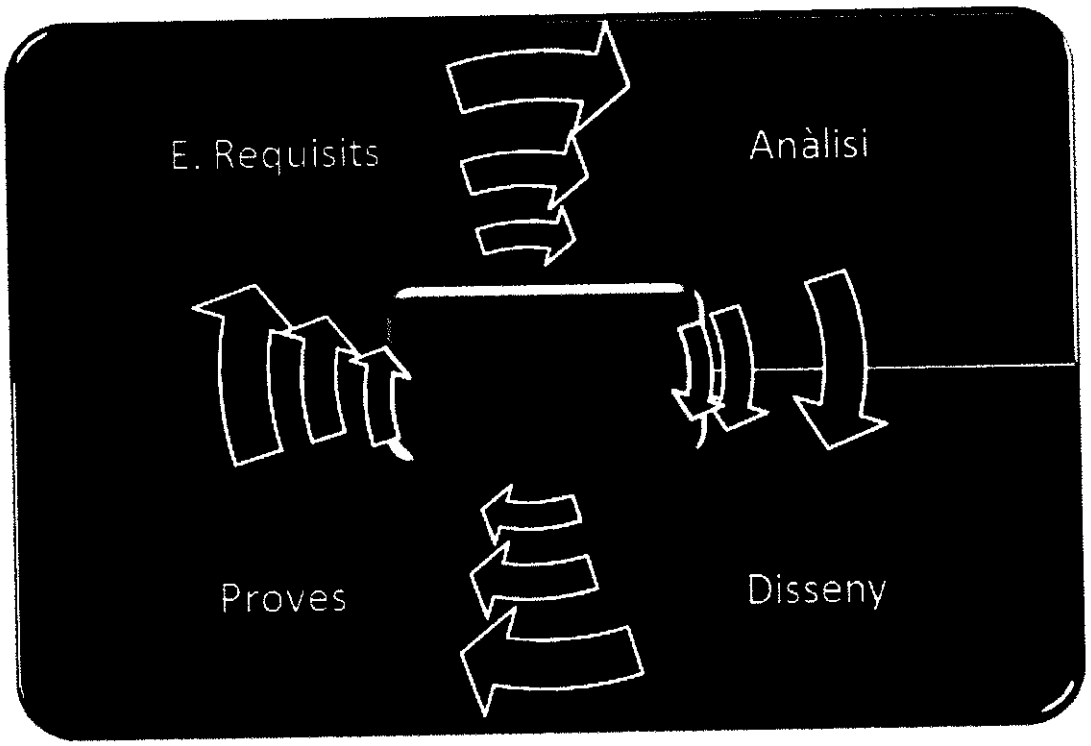

stand_pat = alpha
stand_pat = evaluacio(pos)
retornar beta
Generar Captures
Ordenar Moviments
retornar alpha
valorActual = quiesce Recursiu aplicant el moviment
alpha = max(alpha,valorActual)

6. Planificació i valoració econòmica del projecte

L'objectiu d'aquest apartat és el de presentar la distribució temporal inicial que es va fer de les tasques i la valoració econòmica estimada de les mateixes. Començarem amb un punt d'introducció on s'explicarà la metodologia utilitzada per desenvolupar el projecte. A continuació presentarem la distribució temporal de les iteracions i les seves etapes acabant finalment amb la valoració econòmica. Tots els cronogrames i xifres presentats en aquest apartat seran contrastats amb els de l'execució real del projecte per poder detectar i justificar les possibles desviacions que s'hagin produït.

6.1 Metodologia de desenvolupament en espiral

La metodologia escollida per desenvolupar aquest projecte és la coneguda com desenvolupament amb espiral. Aquesta consisteix, a grans trets, en dividir un projecte amb N iteracions(iteració 1, iteració 2, ...) composta cadascuna per M tasques(i.e. presa de requisits, anàlisi, disseny i implementació, proves,...) . Aquesta metodologia s'escull a causa de que es creu que aportarà al projecte una gestió del risc adequada. El següent dibuix il·lustra la distribució de tasques d'aquesta metodologia:

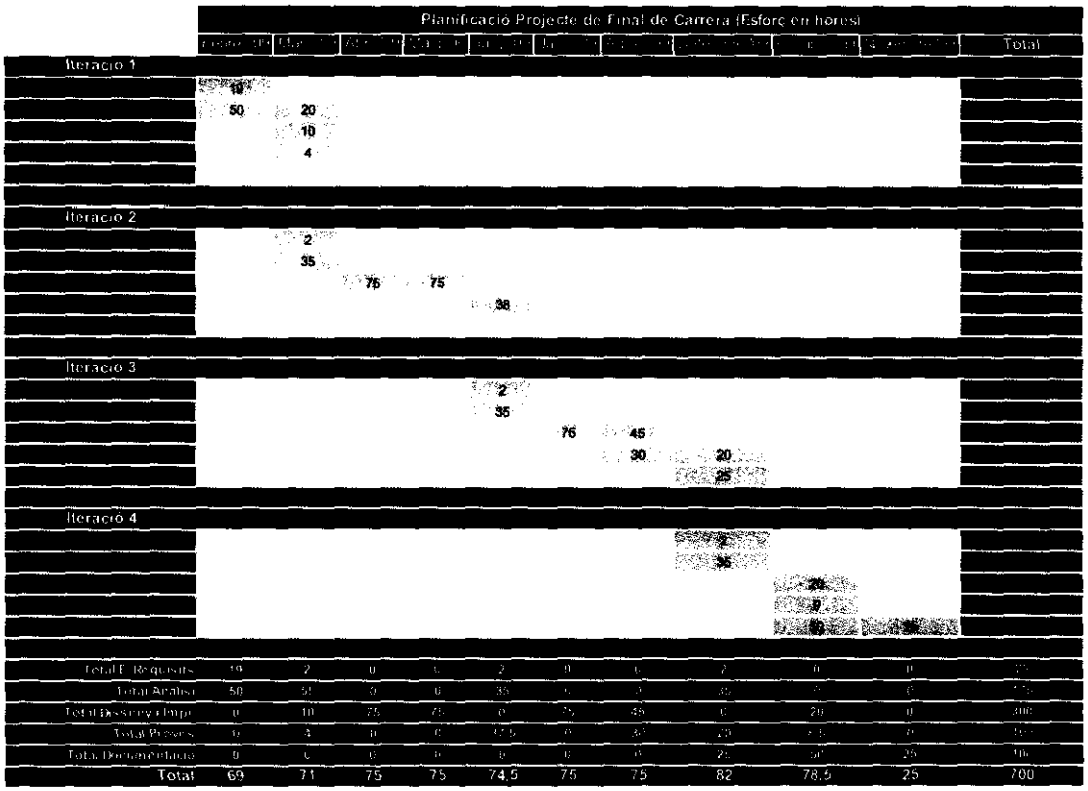


Aquesta metodologia, que és genèrica, s'ha instanciat per al present projecte de la següent manera:

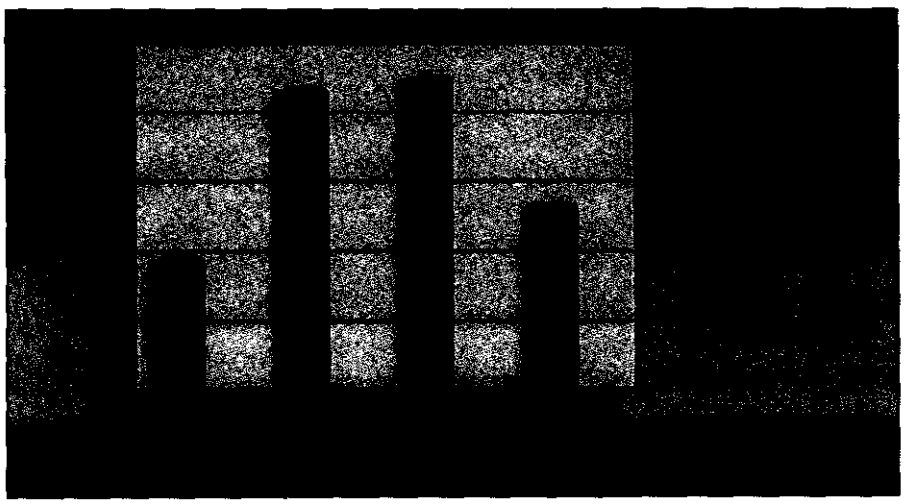
- Iteracions totals : 4
- 2 iteracions curtes(1ª i 4ª)
- 2 iteracions llargues(2ª i 3ª)

6.2 Planificació temporal del projecte

En el cronograma que es presenta a continuació es poden veure les iteracions i tasques de les mateixes distribuïdes en el temps i valorades en esforç(les xifres estan en hores-persona):



Del cronograma anterior, se'n dedueix el següent gràfic resum on es pot veure amb claredat l'esforç relatiu que es dedica a cada tasca dins cada iteració:



6.3 Valoració econòmica del projecte

El primer que necessitem per poder fer la valoració econòmica del projecte és el cost que té la hora de cada rol que es dedicarà al projecte. Per a aquest projecte es proposa la següent tarifa:

		Tarifa €/hora
Rol	Enginyer de Requisits	90
	Analista	70
	Dissenyador/Programador	50
	Beta Tester	50
	Documentador	50

D'aquesta tarifa i del cronograma de l'apartat anterior es dedueix la següent distribució de costos directes:

	Costos de producció estimats Projecte de Final de Carrera (Preus en €)										
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Total
Iteració 1											
	1.710										
	3.500	1.400									
		500									
		200									
Iteració 2											
		180									
		2.450									
			3.750	3.750							
					1.875						
Iteració 3											
					180						
					2.450						
						3.750	2.250				
							1.500	1.000			
								1.250			
Iteració 4											
								180			
								2.450			
									1.000		
									425		
									2.500	1.250	
Total E. Requisits	1710	180	0	0	180	0	0	180	0	0	2250
Total Analisi	3500	3850	0	0	2450	0	0	2450	0	0	12250
Total Dissenyo/Impl.	0	500	3750	3750	0	3750	2250	0	1000	0	15000
Total Proves	0	200	0	0	1875	0	1500	1000	425	0	5000
Total Documentacio	0	0	0	0	0	0	0	1250	2500	1250	5000
Total	5210	4730	3750	3750	4505	3750	3750	4880	3925	1250	39500

A partir de la distribució dels costos directes que acabem de veure i dels costos indirectes aplicables al projecte, arribem a la següent taula on s'hi veuen reflectits els costos totals previstos del projecte:

	Previsió Total de Costos del Projecte (€)										
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Total
Costos Indirectes											
	200	200	200	200	200	200	200	200	200	200	2000
	28	28	28	28	28	28	28	28	28	28	280
	0	0	0	0	0	0	0	0	0	0	0
	300	300	300	300	300	300	300	300	300	300	3000
Costos Directes											
	1.710	180	0	0	180	0	0	180	0	0	2250
	3.800	3.850	0	0	2.450	0	0	2.450	0	0	12250
	0	800	3.700	3.700	0	3.700	2.220	0	1.000	0	15000
	0	200	0	0	1.875	0	1.500	1.000	425	0	5000
	0	0	0	0	0	0	0	1.250	2.000	1.750	5000
Total	5738	5258	4278	4278	5033	4278	4278	5408	4453	1778	44780

Pel que fa al còmput de les despeses dels costos del projecte s'han fet les següents consideracions:

- El director dedicarà al projecte una mitjana de dues hores mensuals
- Els costos de hardware són els corresponents al cost d'un portàtil de 1000€ amortitzat a 36 mesos(vida útil d'un portàtil)
- No existeixen costos de software ja que el projecte es desenvolupa íntegrament amb Open Source
- Els costos de despatx/oficina seran de 300€/mes per desenvolupador

7. Gestió i seguiment del desenvolupament del projecte

En l'anterior apartat hem vist la planificació en temps i diners del projecte. En aquest apartat veurem el cronograma real que s'ha seguit durant el desenvolupament del projecte, les desviacions que s'han produït i l'impacte global que aquestes han tingut de cara als costos del projecte.

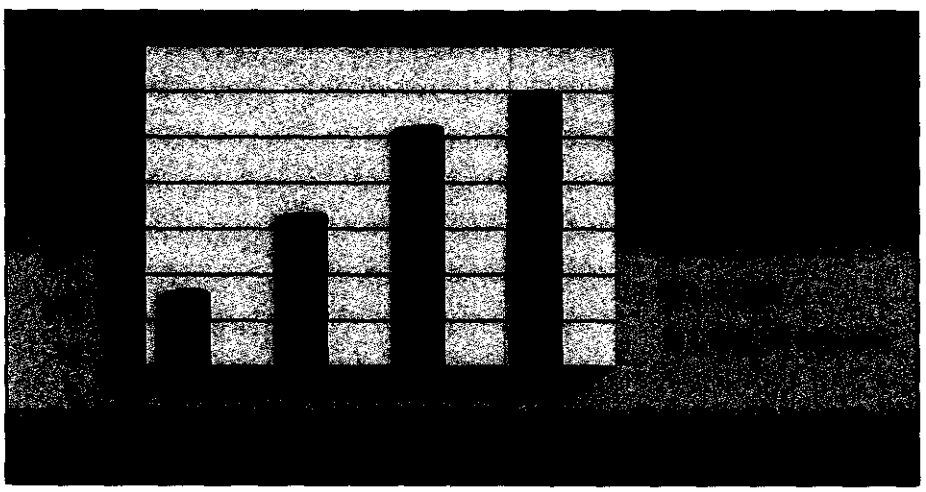
7.1 Cronograma real i desviacions

El cronograma real de l'esforç incorregut en cadascuna de les tasques i iteracions ha sigut el següent:

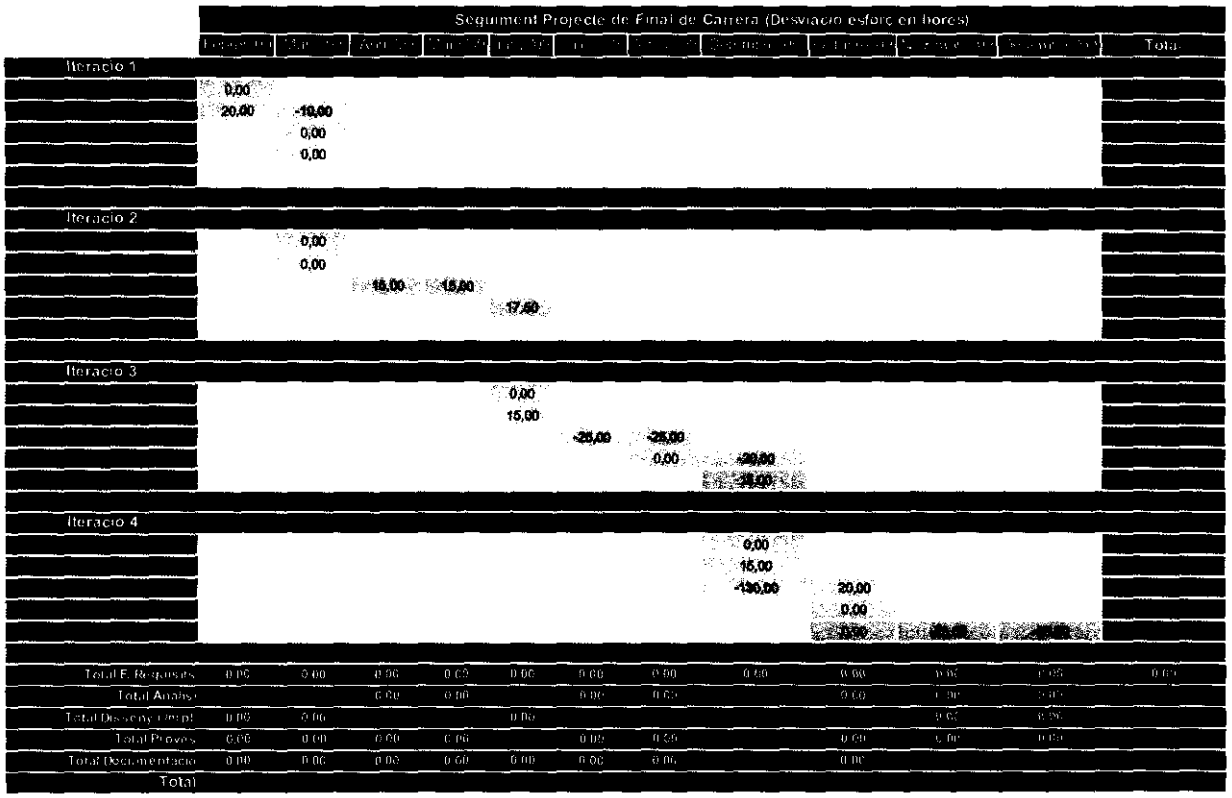
Seguiment Projecte de Final de Carrera (Esforç en hores)

	Març '09	Abril '09	Maig '09	Juny '09	Juliol '09	Agost '09	Setembre '09	Octubre '09	Novembre '09	Desembre '09	Total
Iteració 1											
	19										
	30	30									
		10									
		4									
Iteració 2											
		2									
		35									
			60	60							
					20						
Iteració 3											
					21						
					20						
						100	70				
							30	40			
								10			
Iteració 4											
								2			
								20			
								130	0		
									9		
									50	50	50
Total d'Requisits	49	2	0	0	2	0	0	2	0	0	0	55
Total Analisi	30	65	0	0	20	0	0	20	0	0	0	135
Total Disseny i Impl	0	10	60	60	0	100	70	130	0	0	0	430
Total Proves	0	4	0	0	20	0	30	40	8.5	0	0	102.5
Total Documentacio	0	0	0	0	0	0	0	10	50	50	50	160
Total	49	81	60	60	42	100	100	202	58.5	50	50	852.5

Que presenta la següent distribució de tasques per iteració:

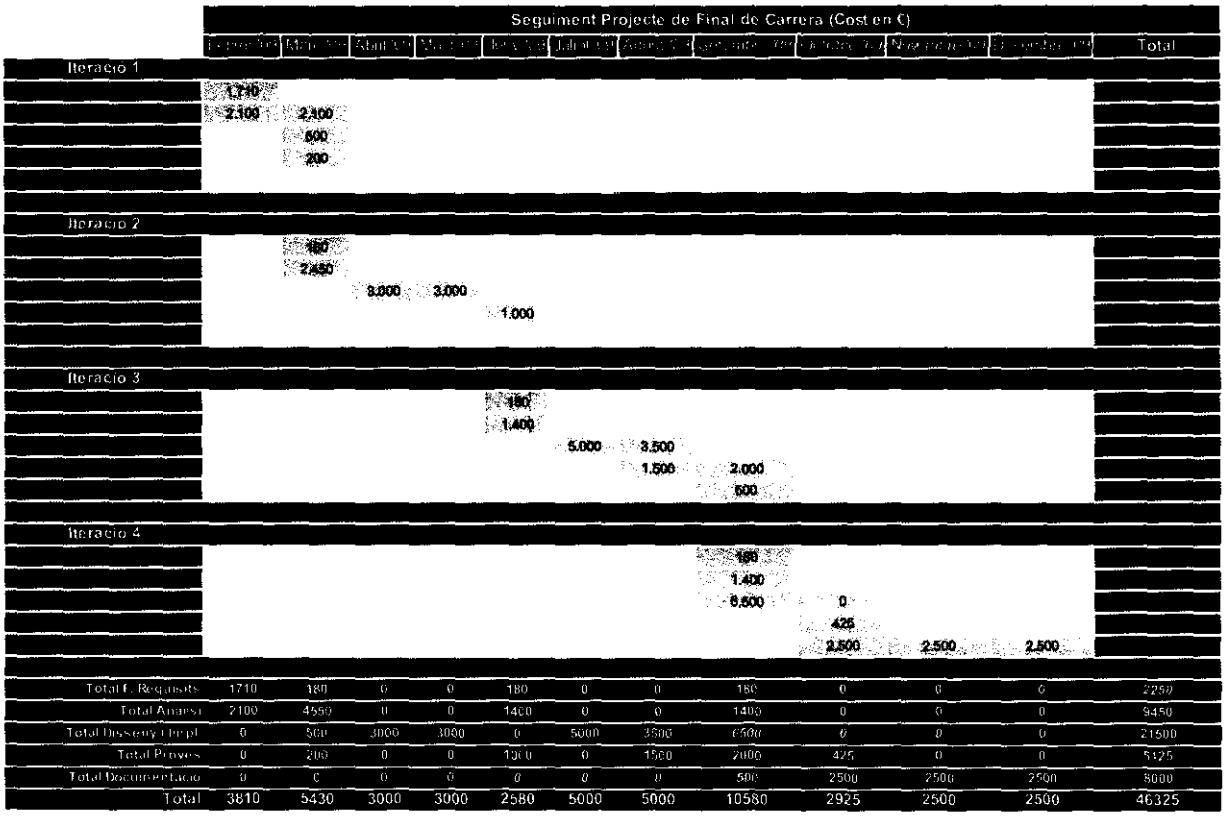


Si creuem el cronograma real amb la planificació inicial del projecte obtenim la següent taula de desviacions en hores:



7.2 Costos reals i desviacions

En el següent quadre es presenten els costos imputables directament a desenvolupament del projecte:



Seguiment Projecte de Final de Carrera (Cost en €)

	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Total
Iteració 1												
	1710											
	2.100	2.200										
		600										
		200										
Iteració 2												
		180										
		2.450										
			3.000	3.000								
					1.000							
Iteració 3												
					180							
					1.400							
						5.000	3.500					
							1.500	2.000				
								600				
Iteració 4												
								180				
								1.400				
								8.500	0			
									425			
									2.500	2.500	2.500	

Total E. Requisits	1710	180	0	0	180	0	0	180	0	0	0	2250
Total Anàlisi	2100	4550	0	0	1400	0	0	1400	0	0	0	9450
Total Disseny i Impl	0	600	3000	3000	0	5000	3500	6500	0	0	0	21500
Total Proves	0	200	0	0	1000	0	1500	2000	425	0	0	5125
Total Documentació	0	0	0	0	0	0	0	500	2500	2500	2500	8000
Total	3810	5430	3000	3000	2580	5000	5000	10580	2925	2500	2500	46325

En la següent pàgina es presenta un quadre resum de les desviacions que s'han produït durant el desenvolupament d'aquestes tasques(en esforç i en diners)

	Seguiment Projecte de Final de Carrera							
	Esforç (Hores)				Cost (€)			
	Planificat	Real	Desv. Abs.	%Desv.	Planificat	Real	Desv. Abs.	%Desv.
Iteració 1								
Iteració 2								
Iteració 3								
Iteració 4								
Total E. Requisits	25	25	0	0.00%	2250	2250	0	0.00%
Total Analisi	175	135	40	22,86%	12250	9450	2800	22,86%
Total Disseny i Impl.	300	430			15000	21500		
Total Proves	100	103			5000	5125		
Total Documentacio	100	160			5000	8000		
Total	**700**	**853**			**39500**	**46325**		

Sumant als costos directes els indirectes obtenim la següent taula amb els costos totals del projecte:

	Costos Reals del Projecte (€)											
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Total
Costos Indirectes												
	200	200	200	200	200	200	200	200	200	200	200	2200
	28	28	28	28	28	28	28	28	28	28	28	308
	0	0	0	0	0	0	0	0	0	0	0	0
	300	300	300	300	300	300	300	300	300	300	300	3300
Costos Directes												
	1.710	180	0	0	180	0	0	180	0	0	0	2250
	2.100	4.950	0	0	1.400	0	0	1.400	0	0	0	9450
	0	500	3.000	3.000	0	5.000	2.500	8.500	0	0	0	21500
	0	200	0	0	1.000	0	1.500	2.000	425	0	0	5125
	0	0	0	0	0	0	0	500	2.500	2.500	2.500	8000
Total	4338	5958	3528	3528	3108	5528	5528	11108	3453	3028	3028	52133

Creuant aquestes dades amb el pressupost inicial que havíem fet ens surten les següents desviacions:

	Desviacions de costos respecte el planificat (€)											
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Total
Costos Indirectes												
	0	0	0	0	0	0	0	0	0	0	200	
	0	0	0	0	0	0	0	0	0	0	-28	
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	-300	
Costos Directes												
	0	0	0	0	0	0	0	0	0	0	0	0
	1.400	-750	0	0	1.050	0	0	1.050	0	0	0	2800
	0	0	750	750	0	-1.250	-1.250	-6.800	1.000	0	0	
	0	0	0	0	875	0	0	-1.000	0	0	0	
	0	0	0	0	0	0	0	750	0	-1.250	-2.500	
Total	1400		750	750	1925				1000			

I, agrupant tots els costos que hem vist a l'apartat de pressupost amb les dades reals d'aquest apartat, ens surt la següent taula resum de costos i desviacions:

	Costos globals del Projecte(€)			
	Pressupost	Real	Desviació	% Desv.
Costos Indirectes				
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Costos Directes				
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Total	44780	52133		

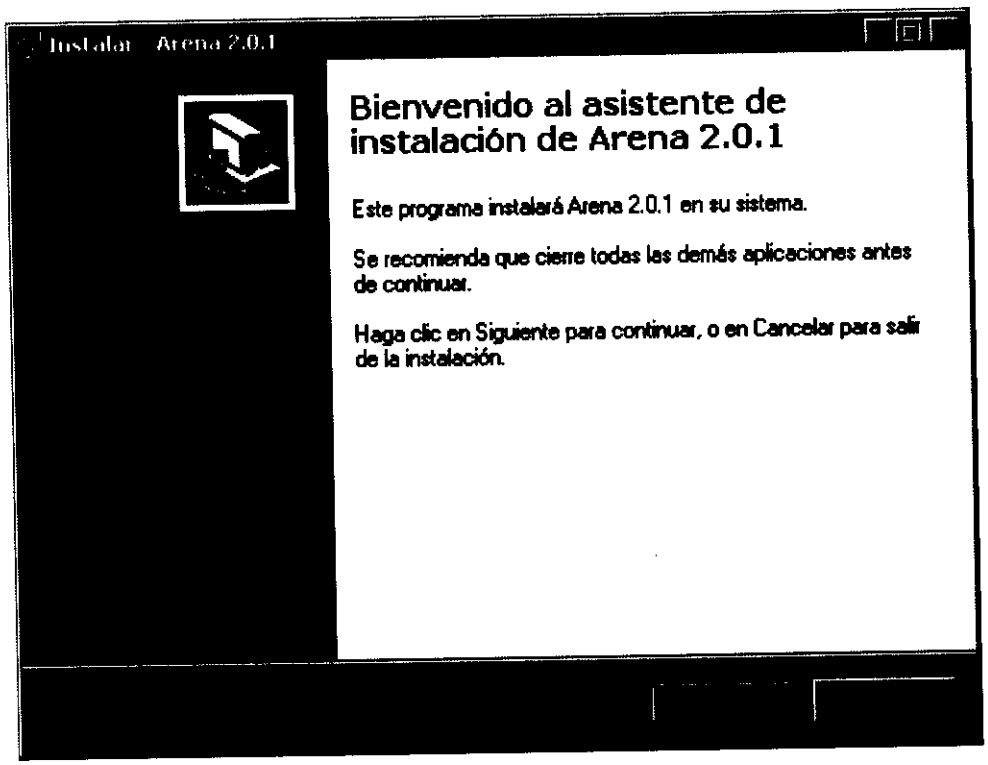
8. Manual d'instal·lació

En aquest apartat es descriuran els passosa seguir per instal·lar el software desenvolupat durant aquest projecte. Aquest procés s'ha dividit en dues tasques: la instal·lació de la interfície gràfica(software de tercers) i la instal·lació del motor d'escacs(software desenvolupat).

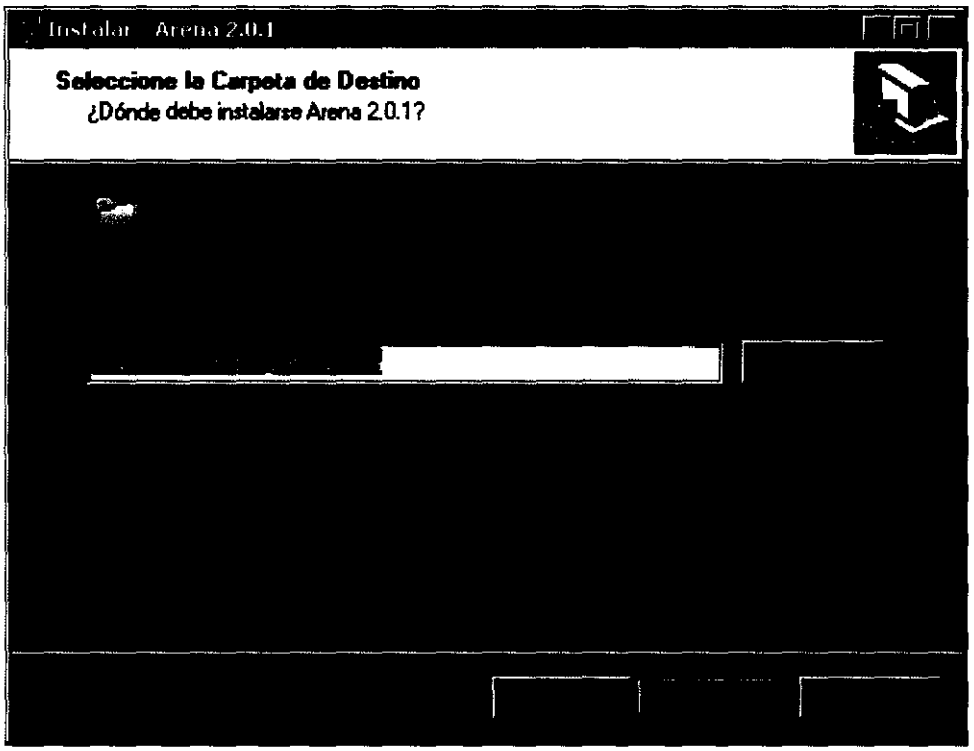
Pel que fa al sistema operatiu, aquí es descriu la instal·lació en Windows encara que, donat que el codi font és totalment portable, el sistema també és instal·lable en altres sistemes operatius com per exemple Linux.

8.1 Instal·lació de la interfície gràfica

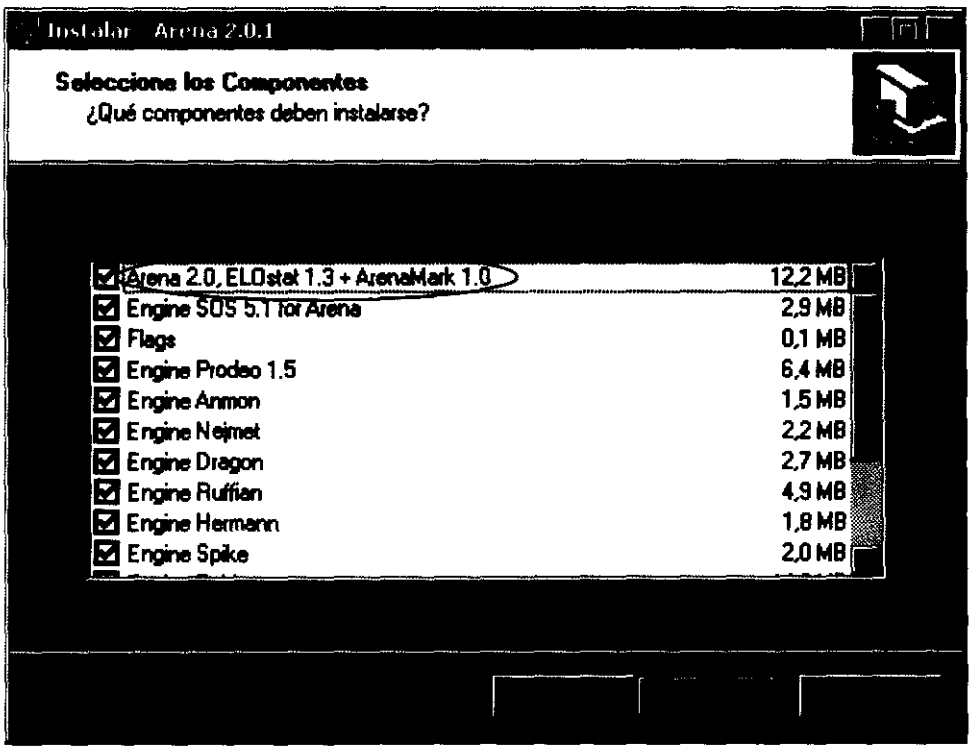
De cara a oferir un exemple complet d'instal·lació, es descriu aquí la instal·lació de la interfície gràfica gratuïta Arena encara que en teoria és possible instal·lar el motor en qualsevol interfície gràfica que implementi el protocol UCI com per exemple les del fabricant ChessBase® o les del fabricant Chess Informant®.



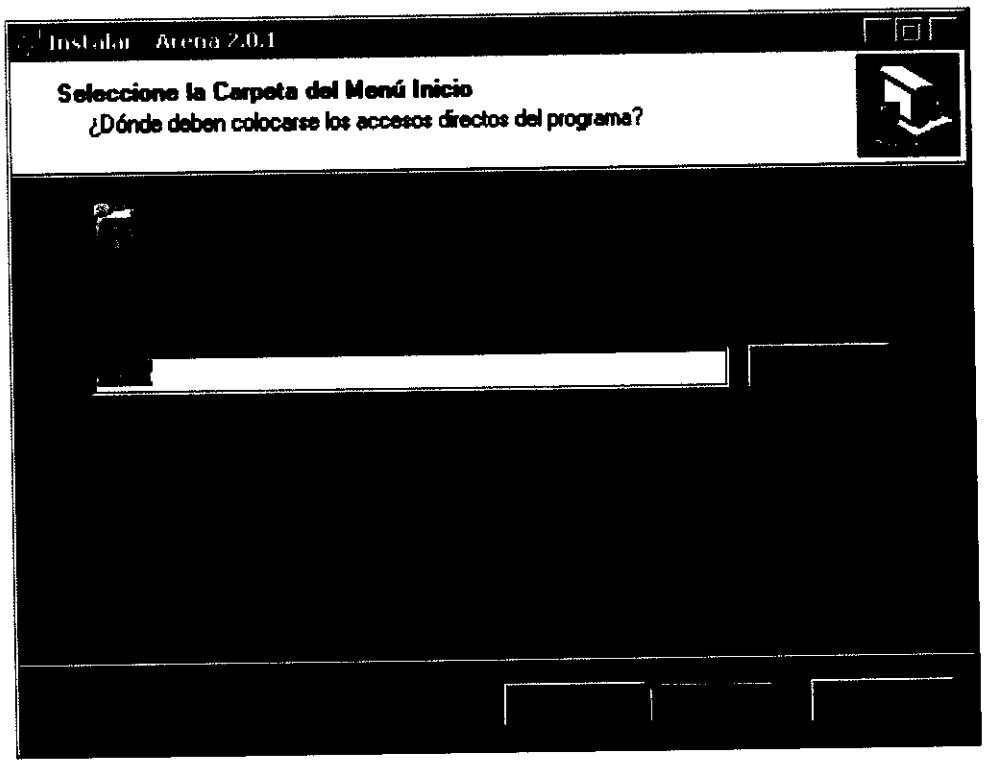
Un cop en la pantalla de benvinguda, senzillament fem click al botó "Siguiente" i passem a la pantalla que es mostra a continuació.



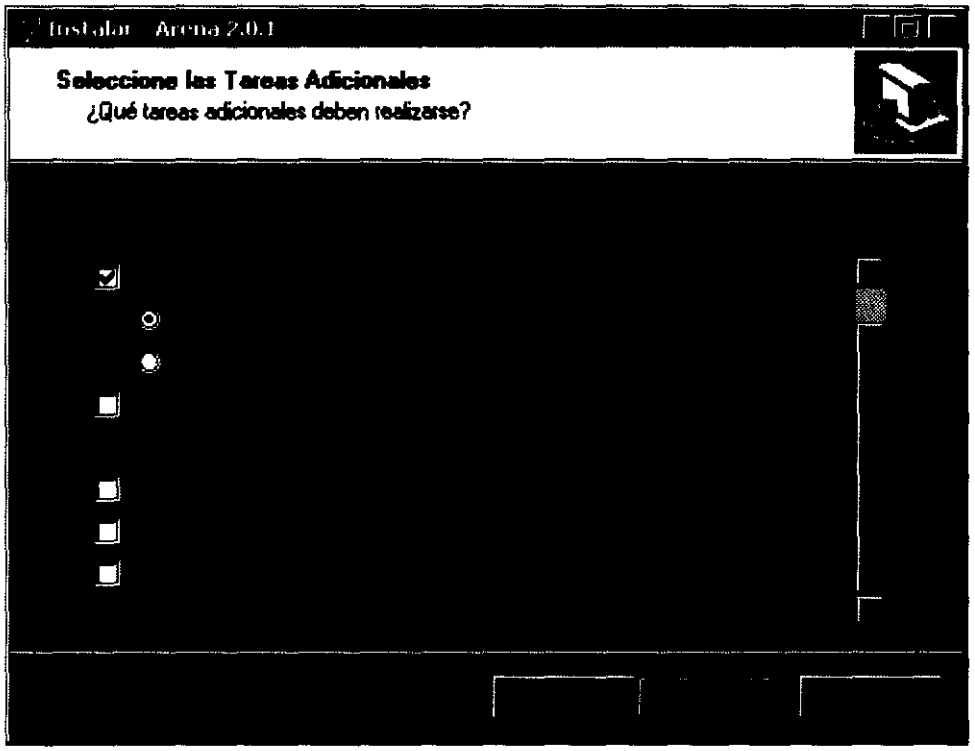
L'objectiu d'aquesta pantalla és el de seleccionar el directori d'instal·lació. L'usuari té la llibertat aquí de seleccionar el directori que més li convingui. A partir d'aquest punt, ens referim al directori seleccionat per l'usuari mitjançant la variable lògica <DIRECTORI_INSTALACIÓ>.



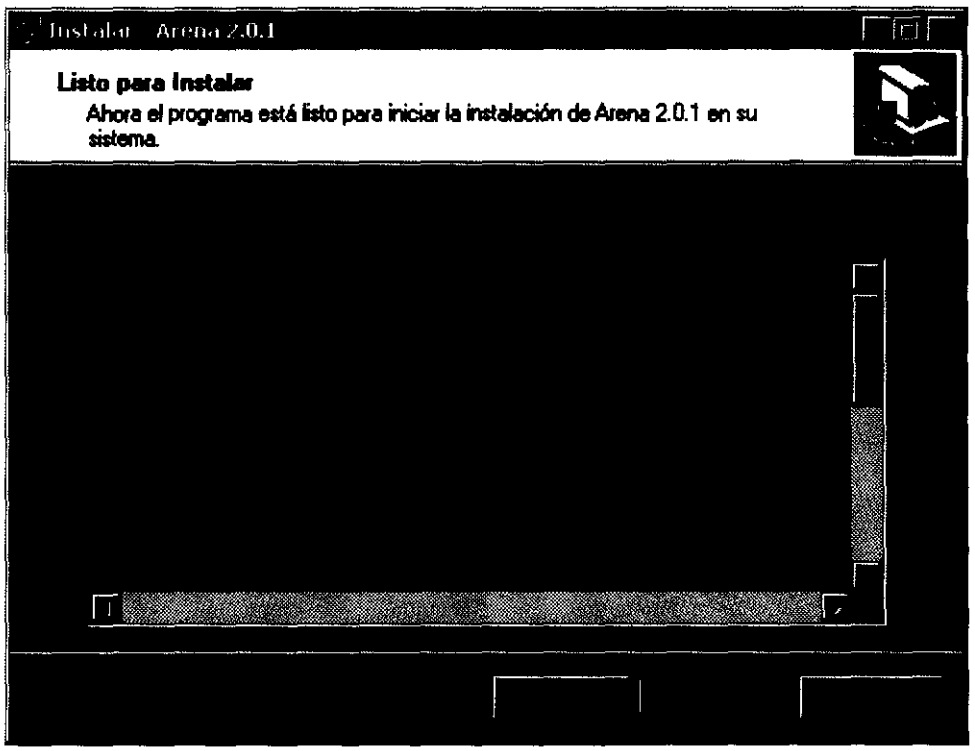
En aquesta pantalla, seleccionem els components a instal·lar. L'únic component estricament requerit és el que s'assenyala amb vermell. Els altres són altres motors de cerca(com el que s'ha implementat en aquest projecte), llibres d'obertures i paquets de llenguatge. L'usuari, a part de la interfície que és requerida, pot seleccionar els components que estimi oportuns. Actualment no està disponible cap versió en català o castellà de la interfície.



Aquesta pantalla permet posar nom i/o canviar la carpeta del menú d'inici de Windows on es col·locarà el llançador de l'aplicació.



Aquesta pantalla ens permet seleccionar quins tipus d'accessos directes es crearan per l'aplicació així com també la tipologia de fitxers que s'hi associaran(i.e. els fitxers pgn són fitxers de texte en un format concret que emmagatzemen una o més partides).



Finalment, fent click al botó *Instalar* instal·lem el programa.

8.2 Instal·lació del motor d'intel·ligència

Un cop instal·lada la interfície gràfica, ens resta només fer la instal·lació del motor i connectar-lo amb la interfície.

8.2.1 Copia dels fitxers del motor

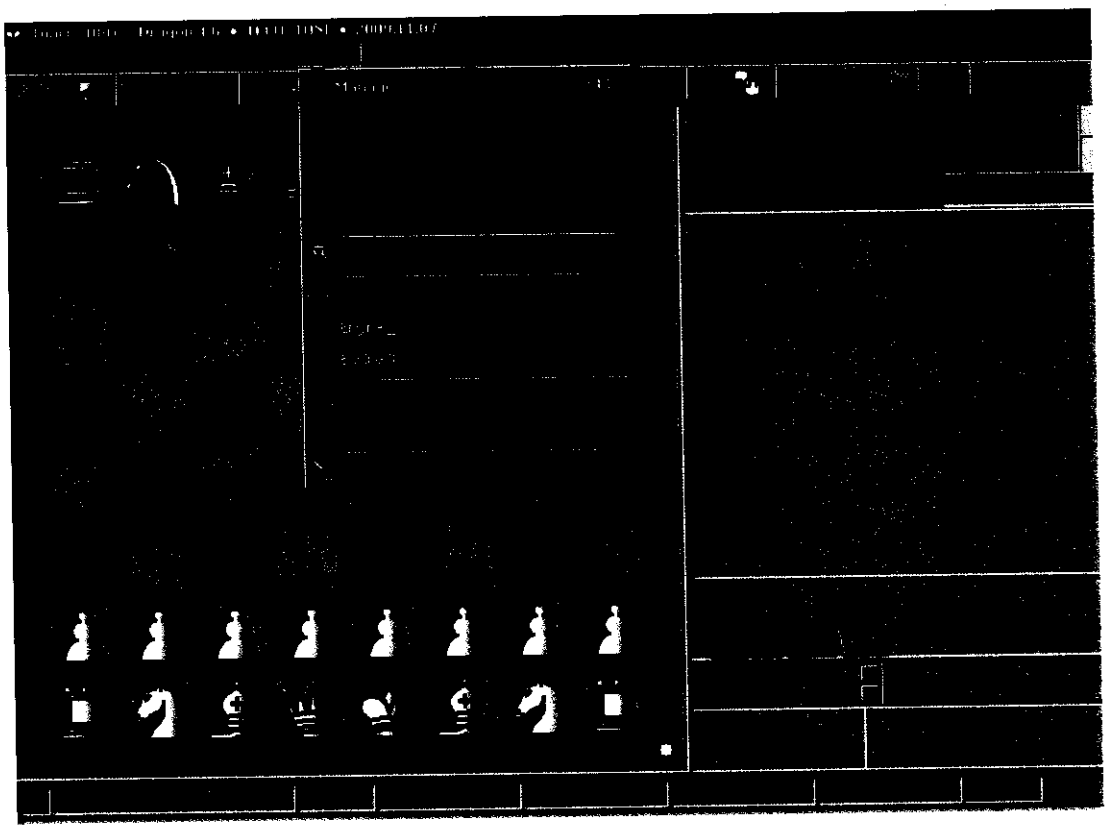
El motor d'escacs es pot copiar en principi en qualsevol carpeta però es recomana fer-ho a la carpeta que la instal·lació ha creat per a tal efecte: *<DIRECTORI_INSTALACIO>\Engines*.

Així doncs, aquesta primera etapa consisteix solament en la còpia de la carpeta *MotorPFC*(present en el CD annexat) al directori *<DIRECTORI_INSTALACIÓ>\Engines*.

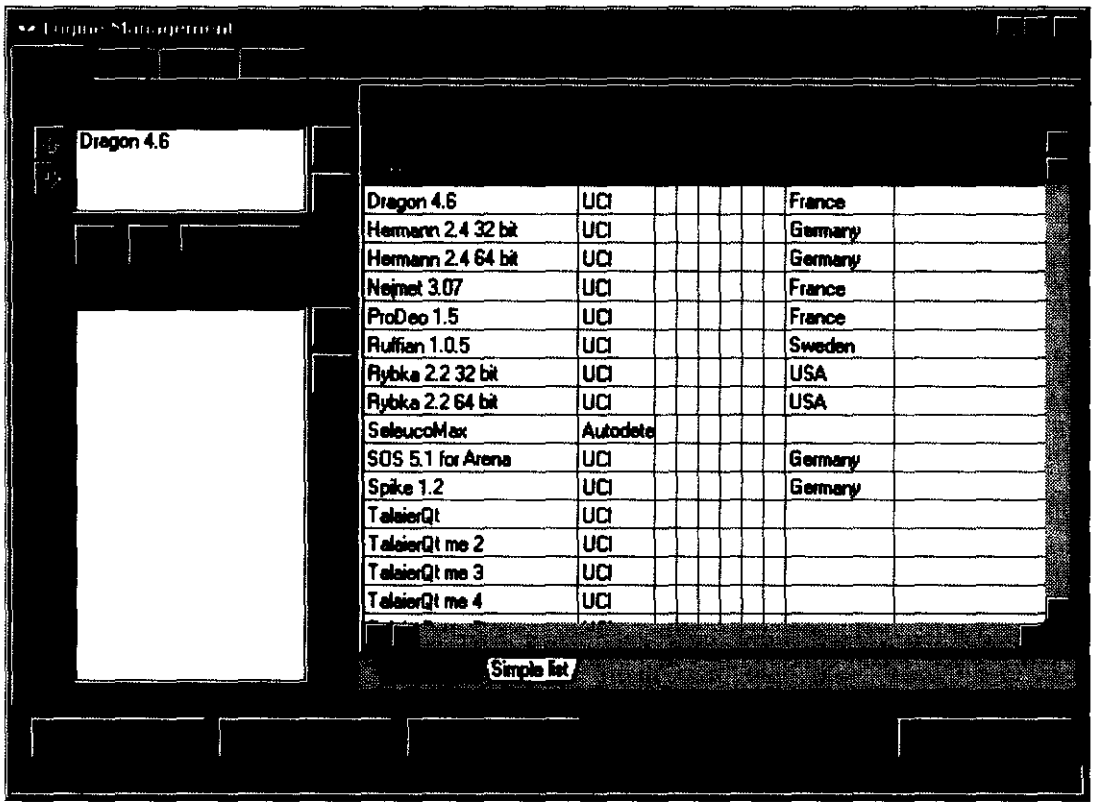
8.2.2 Connexió del Motor a la interfície gràfica

La primera passa a fer en aquesta tasca és obrir la interfície gràfica:

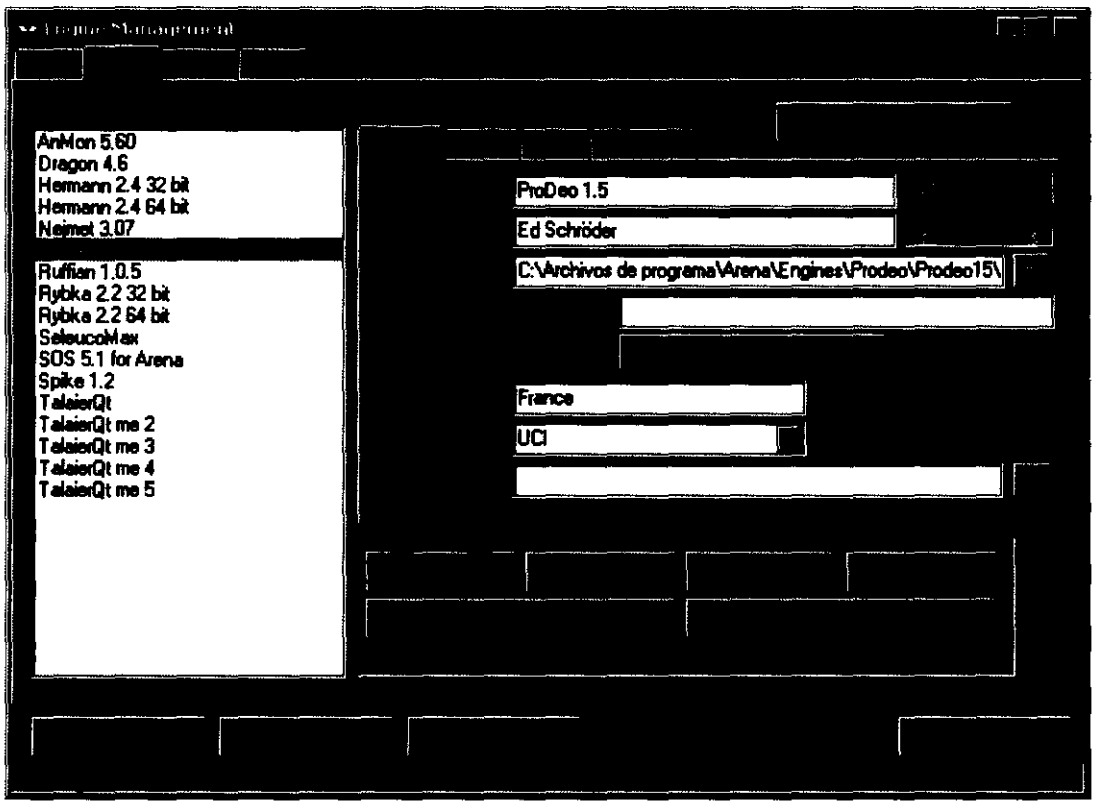
Un cop a dins, seleccionem la opció *Manage* del menú *Engines*:



En el quadre de diàleg que apareix seleccionem la pestanya *Details:*



Per tal d'afegir el motor fem click en el botó *New*:



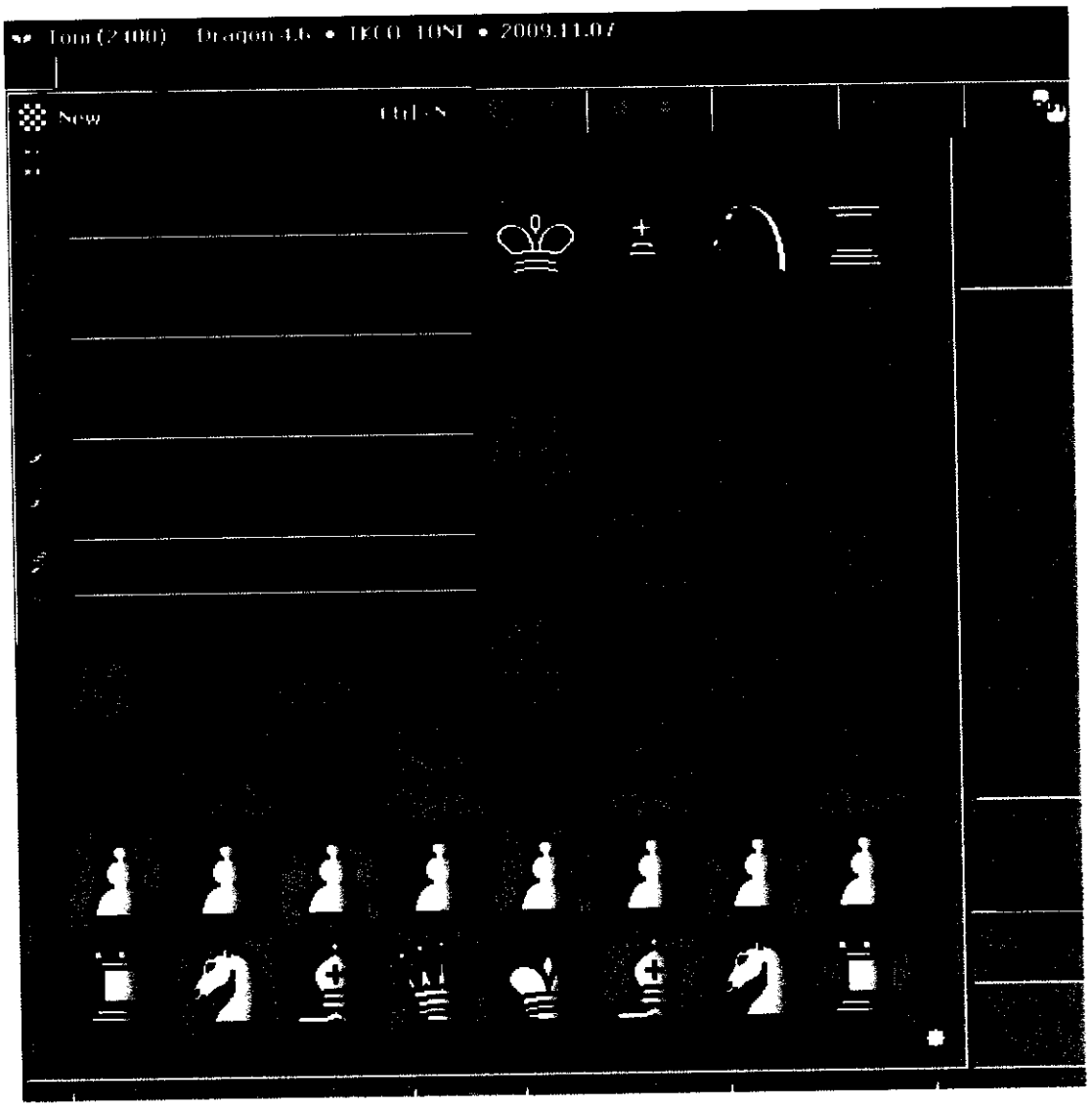
En el diàleg que apareix, seleccionem el fitxer executable contingut a la carpeta que hem copiat al principi i li posem un nom descriptiu al motor com per exemple *MotorPFC*. A continuació fem click a Aceptar i després al botó OK i ja tenim el motor d'escacs connectat a la interfície gràfica.

9. Manual d'usuari

En aquest apartat descriurem les funcionalitats bàsiques de la interfície que ens permetran utilitzar el motor d'intel·ligència desenvolupat.

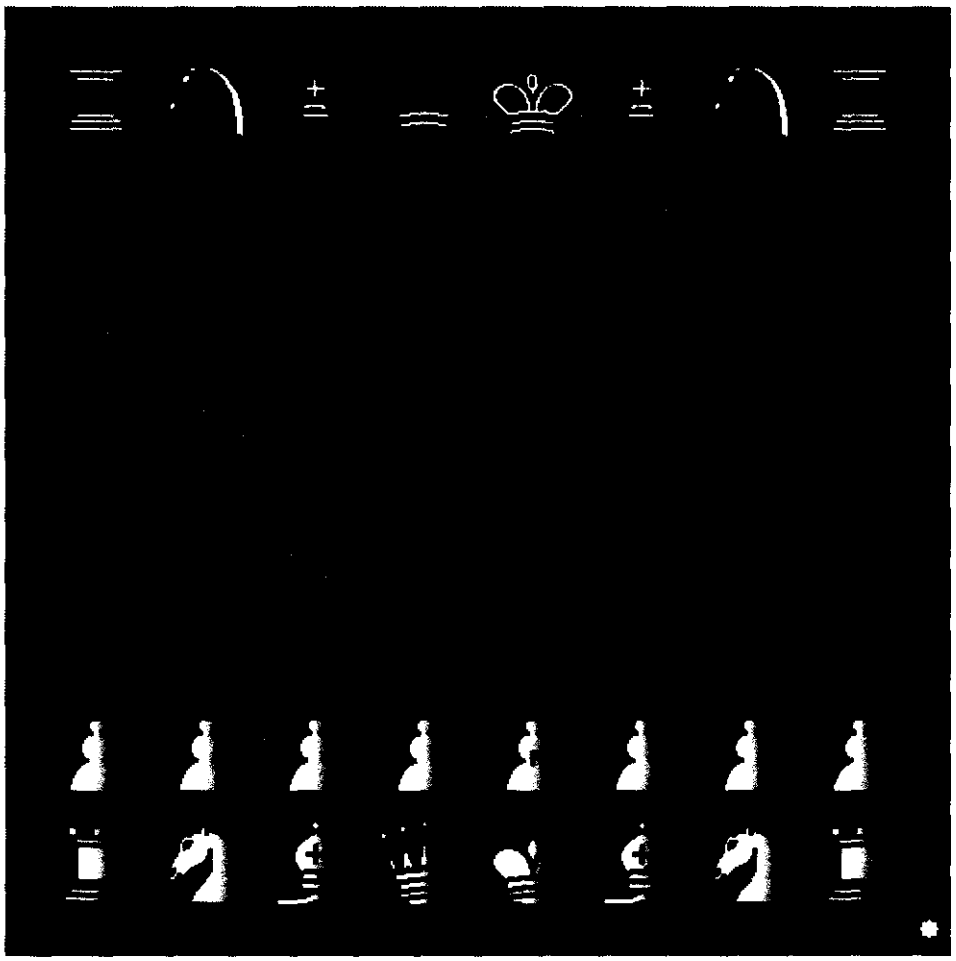
9.1 Nova partida

Per tal de poder començar a jugar, el primer que necessitem és una nova partida. Per crear-la farem click a *File->New*:

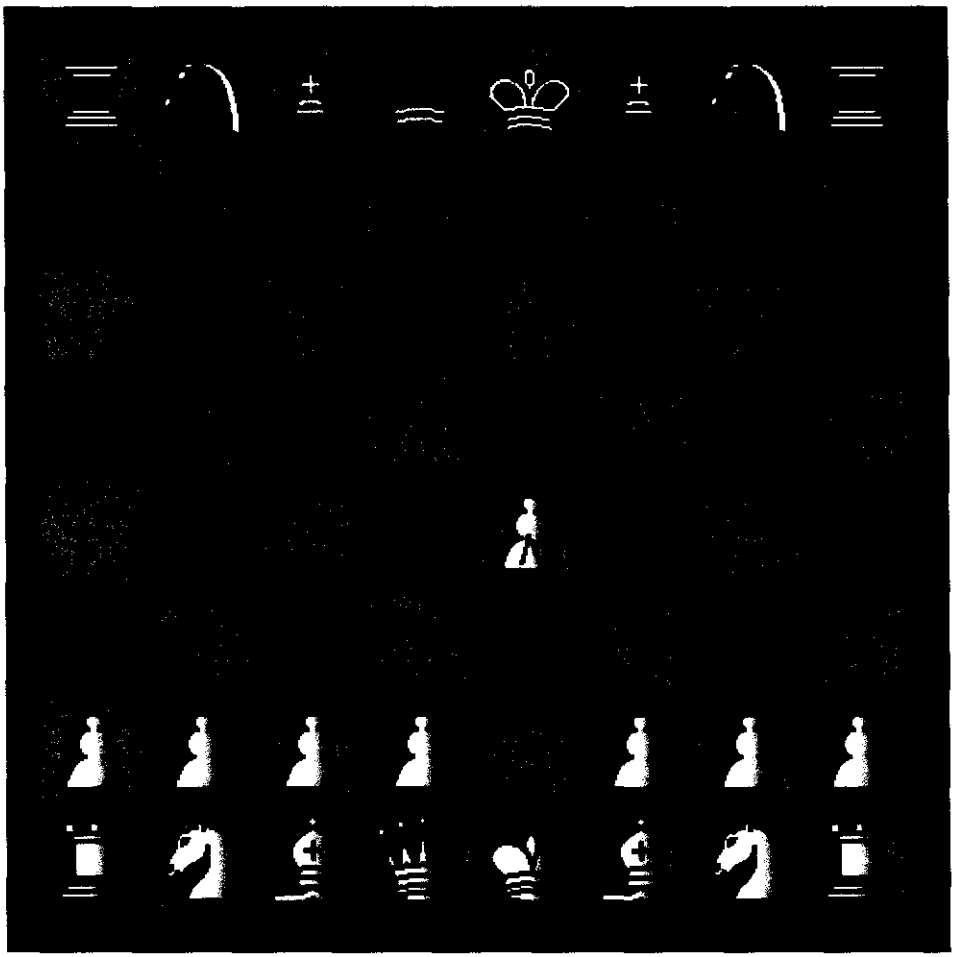


9.2 Introduir jugades

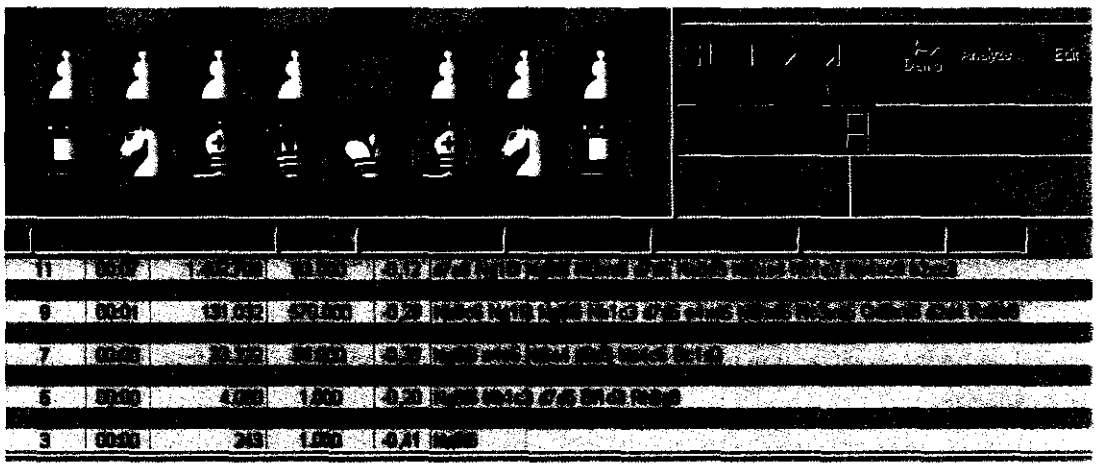
Per introduir jugades en primer lloc assenyalarem la casella on està la peça que volem moure. Aquesta es ressaltarà amb color fúcsia:



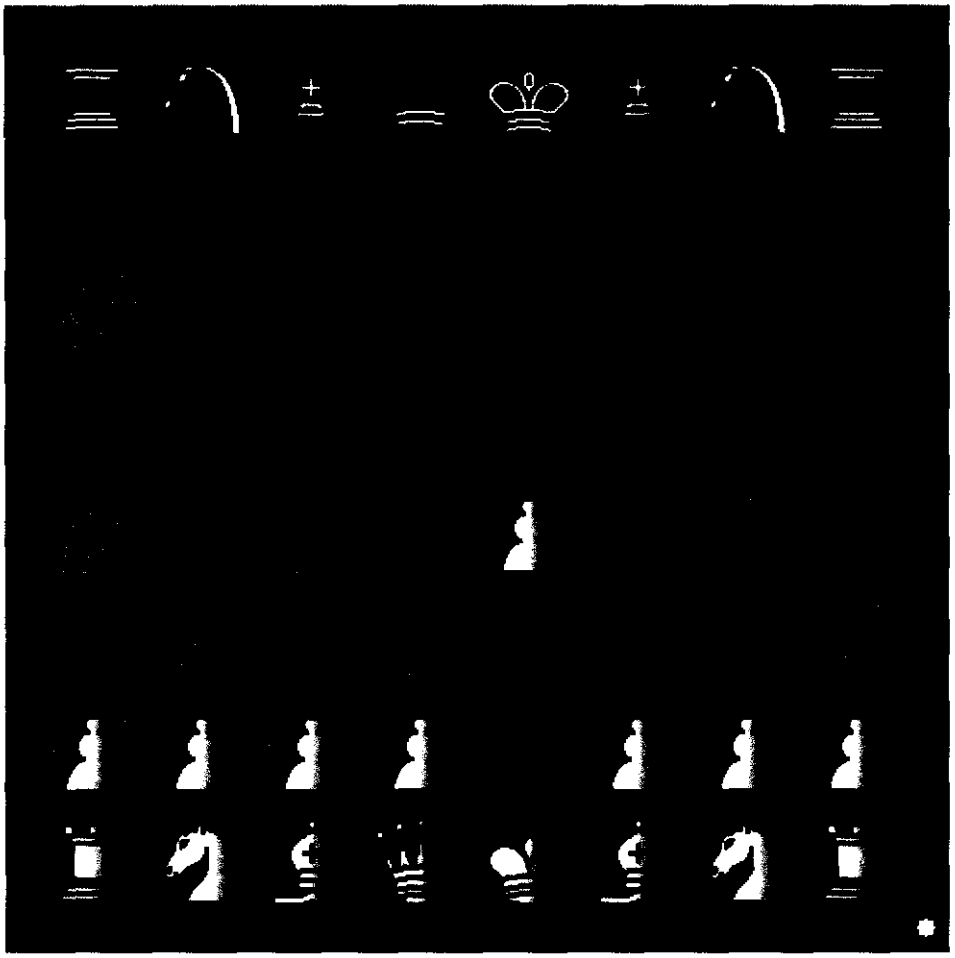
A continuació, per tal d'acabar el moviment farem click a sobre de la casella destí i de seguida la peça que volem moure es traslladarà de casella:



Acte seguit, el motor començarà a calcular la principal línia de joc:



I, finalment, quan s'ha acabat amb el càlcul de la millor jugada aquesta serà efectuada en el tauler cedint un altre cop el torn al jugador humà:



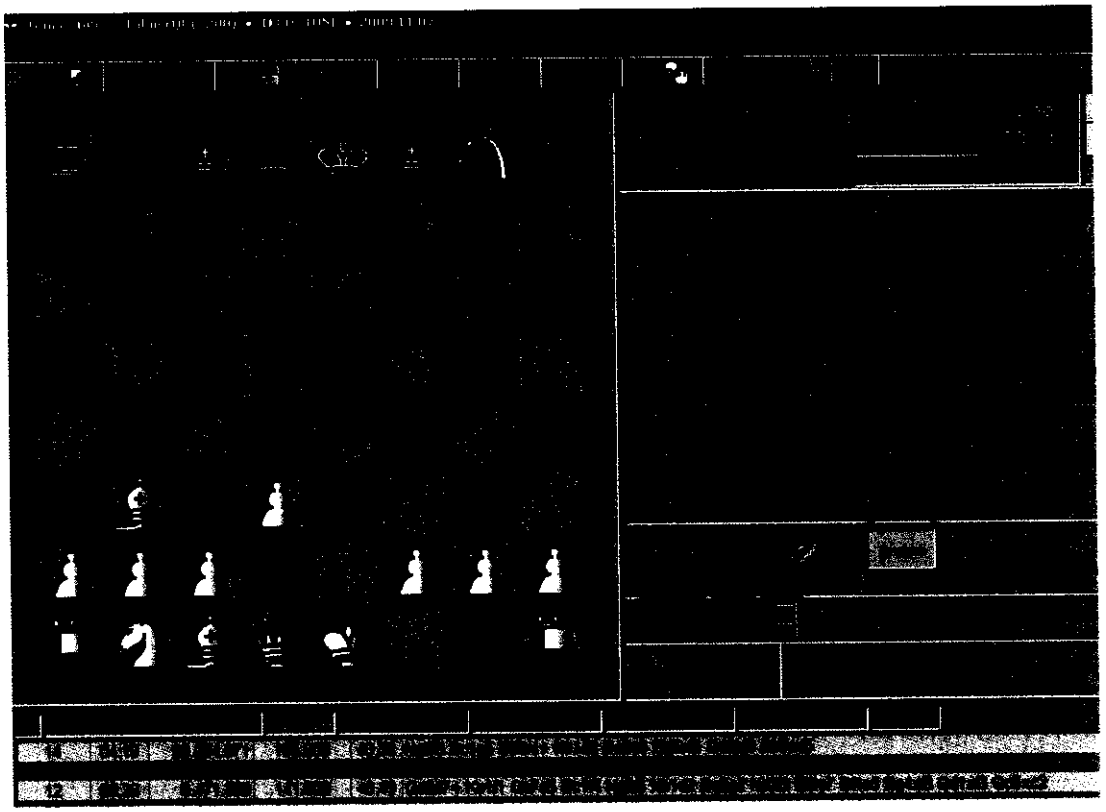
En el cas de que s'hagi seleccionat una casella d'origen equivocada, sempre es pot tornar enrere fent click un altre cop a la casella(el contorn de color fúcsia es desactivarà).

Nota: Quan es comença una nova partida, per defecte el jugador humà juga amb blanques, per tal de cedir el color blanc a la màquina i jugar amb les peces negres serà suficient fer click en el menú *Game->Move Now!*

9.3 Analitzar partida

Aquesta funcionalitat permet analitzar partides que s'han jugat per tal de conèixer, per exemple, quines errades s'han comès.

Per tal de poder analitzar una partida s'ha de seleccionar el mode d'anàlisi infinit mitjançant el botó *Analyse*.



A partir d'aquest moment el motor d'intel·ligència comença a calcular, de manera infinita segons l'algorisme de profunditat iterativa, la millor jugada des de la posició actual i va ensenyant la variant principal en el panell inferior. Des de la posició actual l'usuari pot retrocedir una jugada, avançar una jugada(si és que li queden jugades per avançar) o crear una variant.

10. Conclusions

Els objectius d'aquest apartat són el següents:

- Posar de manifest i sintetitzar els coneixements adquirits durant el desenvolupament del projecte
- Descriure el grau d'assoliment dels objectius
- Identificar, a posteriori, quines han estat les claus de l'èxit d'aquest projecte i quins errors no es tornarien a repetir.

Portar a terme aquest projecte m'ha ajudat a aprofundir i ampliar els meus coneixements en les següents àrees:

- **Intel·ligència artificial:** Tots els fonaments teòrics necessaris provenen d'aquesta disciplina gràcies a la qual es coneixen tots els algorismes de cerca involucrats en la presa de decisions en jocs amb adversari
- **Coneixements específics del domini dels escacs:** Per tal de desenvolupar l'heurística necessària per a l'avaluació de posicions m'ha calgut ampliar els meus coneixements posicionals del joc.
- **Coneixements tècnics:** El fet d'haver d'implementar un software complex, com és un motor d'intel·ligència artificial per els escacs, m'ha portat a ampliar substancialment els meus coneixements tècnics específics del llenguatge C++ i moltes tecnologies perifèriques que he necessitat utilitzar com per exemple llibreries multi-thread o llibreries d'automatització de proves unitàries.
- **Enginyeria del software:** Desenvolupar un projecte de manera integral m'ha portat a haver de dominar i practicar totes les etapes involucrades en aquesta disciplina com per exemple l'anàlisi de requisits o la planificació del projecte.

Pel que fa a l'assoliment dels objectius, es pot dir que s'han assolit tots:

- **Objectiu 1, estudi de les tècniques utilitzades en l'actualitat:** Tot i que no es un objectiu mesurable de manera quantitativa, podem dir que l'objectiu s'ha assolit ja que el projectista coneix en l'actualitat la majoria de tècniques utilitzades en els software més potents de domini públic.(Pot ser que hi hagin coneixements no publicats utilitzats per els softwares comercials).
- **Objectiu 2, implementació d'un programa que jugui al nivell de 1900 ELO:** Aquest objectiu és mesurable i s'ha complert. De 50 partides de prova jugades en el servidor freechess.org contra jugadors amb un elo entre 1800 i 1900 , el software desenvolupat n'ha guanyat 48 i n'ha empatat dues.
- **Objectiu 3, implementació del protocol UCI:** També assolit. Tota la funcionalitat del software es pot utilitzar íntegrament des de qualsevol interfície gràfica que implementi el protocol UCI.

En tercera instància, les claus de l'èxit que he anat identificant durant el transcurs del projecte són les següents:

- Planificació del projecte i replanificació tant bon punt es detecta una desviació.
- Seguiment del projecte intentant complir les fites proposades de manera fidedigne.
- Prototipatge en cada iteració per identificar riscos potencials.
- Documentar-se bé sobre les tècniques ja inventades evitant reinventar la roda.
- Utilitzar una bona metodologia d'Enginyeria del Software .
- Documentar el software durant tot el projecte evitant deixar-ho per el final.
- Utilització de Software Open Source.

Ja per acabar, hem resta comentar aquells errors comesos que crec que hauria d'evitar repetir:

- Compaginar el desenvolupament alhora de dos projecte de mota envergadura.
- Valorar un projecte sense basar-me amb algun antecedent(encara que no sigui un projecte idèntic).
- No recaptar informació suficient quan s'està provant el software.
- Començar un projecte des de zero: molt millor aprofitar codi Open Source i centrarse en la innovació.

11. Glossari

- 1.- ELO: Puntuació que atorguen les federacions als jugadors d'escacs en funció dels resultats en les seves partides que serveix després per elaborar rankings.
- 2.-UCI(Universal Chess Interface): Protocol estàndard de comunicació que implementen algunes interfícies i motors d'escacs que amb l'objectiu de que aquests dos components siguin substituïbles i reutilitzables.
- 3.-Caiguda de bandera: Expressió utilitzada per fer referència al fet de que s'acabi el temps d'un jugador. El jugador al qual se li acaba el temps, perd automàticament la partida.
- 4.-Mode d'anàlisi que ofereixen la majoria d'interfícies gràfiques que permet analitzar durant un temps indefinit a priori una posició donada actualitzant contínuament per pantalla el resultat de la cerca. Aquest mode té sentit quan el motor de cerca ha implementat l'algorisme de profunditat iterativa i permet aprofundir en aquelles posicions que no una puntuació clara.
- 5.-Línia/Variant Principal: Seqüència òptima de jugades assumint un joc perfecte per part dels dos rivals respecte d'una funció d'avaluació donada.
- 6.-Anàlisi retrospectiu: Anàlisi que es fa a una partida d'escacs un cop jugada amb la finalitat de detectar errades en les jugades que ha fet cadascun dels dos jugadors.
- 7.-Taules per ofegat: Empat que es dona quan un dels jugadors no té cap jugada legal a la seva disposició i el seu rei no es troba en escac.
- 8.- Motor d'escacs: Part d'un software d'escacs encarregada de proporcionar tota la funcionalitat relacionada amb la intel·ligència del joc.
- 9.-Chaturanga: Joc antic de batalla estratègica considerat antecessor dels escacs.
- 10.-Escac i mat: Objectiu final d'una partida d'escacs. Quan s'amenaça al rei rival i aquest no pot fugir de l'amenaça es diu que s'ha donat escac i mat al rival i aquest guanya la partida.
- 11.-Final de partida: Fase terminal del joc caracteritzada per la presència de poques peces en el tauler provocant sovint que el desenllaç de la partida sigui clar si no es cometen errades.
- 12.- Zugzwang: Situació atípica que es dona en algunes partides en la qual el jugador que té el torn tindria una posició millor si li toqués moure al rival
- 13.- Peons doblats: Situació en la qual dos peons del mateix color ocupen una mateixa columna. Es considera una debilitat estratègica a llarg plaç a causa de la reduïda mobilitat d'aquesta peça.
- 14.- Obertura: Fase inicial de la partida on hi han encara la majoria de peces en joc. És la fase més incerta del joc i en la qual el càlcul que realitzen els programes resulta de menys utilitat.
- 15.-FEN(Forsyth–Edwards Notation): Notació estàndard que permet representar de manera compacte posicions d'escacs arbitràries. Fou utilitzada inicialment per jugar partides per correspondència i en l'actualitat l'utilitza el protocol UCI per descriure posicions.

12. Bibliografia

[1] Claude Shannon (**1950**). *Programming a Computer for Playing Chess*, Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950

[2] A. G. Bell (**1972**). *Games Playing with Computers.*

[3] D.F. Beal (**1999**). *The Nature of MINIMAX Search* ISBN 90-62-16-6348

[4] http://wbec-ridderkerk.nl/html/UCIProtocol.html

[5] Hyatt, R.M. (**1984**). *Using Time Wisely.* ICCA Journal, Vol. 7 ISSN 0920-234X.

[6] Richard E. Korf (**1985**). *Depth-first Iterative-Deepening: An Optimal Admissible Tree Search*

[7] Russell, Stuart J.; Norvig, Peter (2003), *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, NJ: Prentice Hall , ISBN 0-13-790395-2

[8] Pearl, J. (1980). *Asymptotic Properties of Minimax Trees and Game-Searching Procedures.* Artificial Intelligence ISSN 0004-3702.

[9] Pearl, J. (**1980**). *Scout: A Simple Game-Searching Algorithm with Proven Optimal Properties.* Proceedings of the First Annual National Conference on Artificial Intelligence. Stanford.

[10] Finkel, R.A. and Fishburn, J.P. (**1980**). *Parallel Alpha-Beta Search on Arachne.* IEEE International Conference on Parallel Processing

[11] Winands, M.H.M., van de Herik, H.J., Uiterwijk, J.W.H.M., and Werf, E.C.D. van der (**2003**). *Enhanced forward pruning.*

[12] Warnock, T. and Wendroff, B. (**1988**). *Search Tables in Computer Chess.* ICCA Journal, Vol. 11, No. 1. ISSN 0920-234X.

[13] Hyatt, R. M. And Mann, T. (**2002**). *A lock-less transposition table implementation for parallel search chess engines.* ICGA Journal, Vol. 25, No. 1

[14] chessprogramming.wikispaces.com

[15] http://web.archive.org/web/20070822204120/www.seanet.com/~brucemo/topics/hashing.htm

[16] D. Costal, M. Ribera Sancho, E. Teniente, Enginyeria del Software Especificació, Especificació de sistemes orientats a objectes amb la notació UML, Edicions UPC, 3ed **2003**